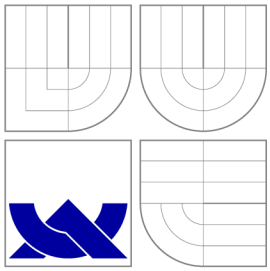
BRNO UNIVERSITY OF TECHNOLOGY

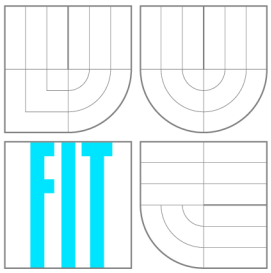
Faculty of Information Technology

PHD THESIS

Brno, 2016 Ing. Eva Zámečníková



BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FORMAL MODEL OF DECISION MAKING PROCESS FOR HIGH-FREQUENCY DATA PROCESSING

FORMÁLNÍ MODEL ROZHODOVACÍHO PROCESU PRO ZPRACOVÁNÍ VYSOKOFREKVENČNÍCH DAT

PHD THESIS
DISERTAČNÍ PRÁCE

AUTHOR
AUTOR PRÁCE
Ing. EVA ZÁMEČNÍKOVÁ

SUPERVISOR
ŠKOLITEL
doc. RNDr. JITKA KRESLÍKOVÁ, CSc.

BRNO 2016

Abstract

This thesis deals with the issue of the processing of high-frequency time series. It primarily focuses on the design of algorithms and methods for support of predicting these data. The result of this work is a model supporting the decision-making process implemented into a complex platform. The model designs the method of formalization of business rules which describes the decision-making process. The designed model must meet the conditions of the robustness, scalability, real-time processing and econometrics requirements. The thesis summarizes the current knowledge and methodologies for the processing of high-frequency financial data which can be found on the stock exchange.

The first part of the work describes the basic principles and approaches currently used in the processing of high-frequency data. The next part deals with the description of an appropriate complex event platform and is subsequently devoted to prediction and data processing itself, using the chosen platform. Emphasis is on selecting and editing a set of rules that controls the decision-making process. The newly designed method describes the set of rules by using matrix grammar. This grammar belongs to the grammars with regulated rewriting and thus it may control the data processing by the defining of the matrices.

Abstrakt

Tato disertační práce se zabývá problematikou zpracování vysokofrekvenčních časových řad. Zaměřuje se na návrh algoritmů a metod pro podporu predikce těchto dat. Výsledkem je model pro podporu řízení rozhodovacího procesu implementovaný do platformy pro komplexní zpracování dat. Model navrhuje způsob formalizace množiny podnikových pravidel, které popisují rozhodovací proces. Navržený model musí vyhovovat splnění požadavků na robustnost, rozšiřitelnost, zpracování v reálném čase a požadavkům ekonometriky. Práce shrnuje současné poznatky a metodologie pro zpracování vysokofrekvenčních finančních dat, jejichž zdrojem jsou nejčastěji burzy.

První část práce se věnuje popisu základních principů a přístupů používaných pro zpracování vysokofrekvenčních časových dat v současné době. Další část se věnuje popisu podnikových pravidel, rozhodovacího procesu a komplexní platformy pro zpracování vysokofrekvenčních dat a samotnému zpracování dat pomocí zvolené komplexní platformy. Důraz je kladen na výběr a úpravu množiny pravidel, které řídí rozhodovací proces. Navržený model popisuje množinu pravidel pomocí maticové gramatiky. Tato gramatika spadá do oblasti gramatik s řízeným přepisováním a pomocí definovaných matic umožňuje ovlivnit zpracování dat.

Keywords

High-frequency data, CEP, time series, business rules, decision-making process, real-time processing, formalization, Esper.

Klíčová slova

Vysokofrekvenční data, CEP, časové řady, podniková pravidla, rozhodovací proces, zpracování v reálném čase, formalizace, Esper.

Reference

ZÁMEČNÍKOVÁ, Eva. *Formal Model of Decision Making Process for High-Frequency Data Processing.* Brno, 2016. PhD thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Kreslíková Jitka.

Formal Model of Decision Making Process for High-Frequency Data Processing

Declaration

I hereby declare that this thesis is my own work that has been created under the supervision of doc. RNDr. Jitka Kreslíková, CSc. Some parts of this thesis are based on consultation on practice with experts. Where other sources of information have been used, they have been duly acknowledged.

.....
Eva Zámečníková
August 30, 2016

Acknowledgements

I would like to thank doc. RNDr. Jitka Kreslíková, CSc. for her support during the supervision of this work, and for her valuable suggestions and recommendations. This work was partially supported by the FR97/2011/G1, CZ.1.05/1.1.00/02.0070, FITS-10-2, and FIT-S-14-2299 grant project "Research and application of advanced methods in ICT".

Contents

Chapter 1

Introduction

This work discusses the issue of the processing high-frequency financial time series. These series are short-term time series that occur at frequencies of lower than a week, and if we consider data from financial markets, then this frequency is determined in hours and much shorter intervals. When monitoring the data in such a high frequency, large volumes of observed data are produced. To process such an amount of data, conventional statistical methods commonly used to describe the time series are not sufficient. For the processing of high-frequency data, nonlinear models are used because linear models are inadequate. While the space of linear models is considered closed and well researched, the space of nonlinear models has as yet been inadequately explored.

The work focuses primarily on the design of algorithms and the methods for the support of prediction of high-frequency data and the choice of platform for their processing. The proposed models must conform to meeting the conditions of robustness, scalability, real-time processing requirements and econometrics. The first part is devoted to the description of the basic principles and approaches currently used for processing high-frequency data. High-frequency data (or financial data) are very variable. At each time unit – tick – new data are generated that indicate the development of the series. Today it is not only important to process data, but also to predict their course, and thus to estimate trends in the data and to find new patterns in the evolution of data.

High-frequency data belong within their properties to the group of Big Data. This is a technical category which can be characterized by using what are called 3V properties – volume, velocity and variety. These data are diversely structured data files whose size is beyond capturing, processing and managing in a reasonable time by using conventional software tools. Big data can be stored into the data warehouse using ETL procedures. These methods include three levels of processing – extraction, transformation and loading of data. For handling of these large data, traditional platforms and traditional methods of storing and processing data are not adequate. Therefore a comprehensive platform and tools based on the "Cloud Computing" model were created. Cloud computing is the sharing of hardware and software resources over the network.

Whole platforms for processing of high-frequency data are available that include complex methods for data preprocessing, analysis and filtering. They also contain basic methods for prediction, based on patterns detected in historical data. For further research, we chose a platform which meets these requirements – the event processing in real time, modularity and independence of the operating system. Part of this platform is an EPA agent (Event Processing Agent) which follows the flow of events and looks for the patterns and responds to them in accordance with pre-defined functions. It takes high-frequency data on the input

and creates new events to the output according to a given set of rules. Agents are divided into three categories: input filters, maps and constraints.

With regard to the basic rules for real-time processing, a suitable platform was selected. For the extraction of rules and patterns from input streams of events and for the data processing itself, the chosen complex platform is used. The following part of the thesis is devoted to the selection of a topic that is relatively unexplored in this area. Because the designed model should work in real time, the resulting model will respect the requirements for data processing at runtime. These requirements are listed in this work.

The main goal of the thesis is to design and implement a decision support system for high-frequency data processing. This system is implemented as a module integrated into the existing solution of a complex event platform.

1.1 Motivation

The motivation was to create method that is based on studies of the latest scientific knowledge and to simultaneously confront with possibilities achievable in real environment with the available tools and technologies. Complex event processing is a rapidly emerging technology. Within the past decade, numerous new platforms and approaches for the processing of high-frequency data were created. In 2002, David Luckham introduced the first concepts and basic theory regarding complex event processing in his monography, *The Power of Events – An Introduction to Complex Event Processing in Distributed Enterprise Systems* in 2002 [40]. In his next monography about complex events, *Event Processing For Business* [41], he mentions the stages in the development of modern event processing in business. According to the author, there are 4 stages recognized in complex event processing (CEP) – Simple CEP (1999–2007), Creeping CEP (2004–2012), CEP as recognized information technology (2009–2020) and Ubiquitos CEP (since 2020).

We have now entered Phase 3, where the expansion of CEP application is expected into many different markets. A new open source development of event processing tools will be formed. The effort to formally describe these systems is connected with this. It is expected that the next generation of high-level languages for specifying complex event patterns and rules will occur. According to David Luckham, the next trends will also be able to formalize and standardize CEP [41]. We decided to devote this thesis to the areas of the decision-making process and to the formalization of the business rules' description.

1.2 Aims of the Thesis

Two main goals of this thesis are:

- The main objective is to design a method for the formalization of business rules which are used during decision making. The decision-making process is a part of complex event processing platforms. These platforms are primarily focused on the processing of high-frequency data from different sources. According to the defined set of business rules, actions are determined which lead to the data prediction.

- The second goal is to investigate and experimentally validate the result of the newly designed model. Experimental results will be measured on the historical set of real data by using the chosen complex event processing platform.

Besides these two goals, this thesis summarizes the techniques and approaches currently used for the processing of financial time series and high-frequency data.

1.3 Structure of the Thesis

The thesis consists of seven chapters and it is organized as follows.

- In the Chapter 1, the reader is informally introduced to high-frequency data and to the motivation and the main goals of this thesis.
- Chapter 2 gives the overview of the methods and fundamental definitions used for the time series processing. There is also given a description of characteristics of economic and high-frequency time series.
- Chapter 3 describes the complex event processing for high-frequency data. These data can be processed by using complex event processing platform and the reference architecture of this platform is characterized. The desription of FOREX, the great source of high-frequency data, follows.
- Chapter 4 is dealing with desicion-making proces in complex event processing. There are named the representative tools for decision managing – decision support systems, business rules and decision tables.
- The following Chapter 5 discusses the design of the business rules formalization by using formal grammar. Main advantages of this approach are named. In second part of this chapter is decribed the Esper engine and basic priciples for the use of this tool.
- In the Chapter 6 the measurements and experimental results can be found. We focused on several parameters which where measured and created a test cases for them. There are described the tests of the method and the parameters which were required as one of the task of this thesis.
- The last chapter, Chapter 7, conclude the main ideas and results of the thesis and discuss possible future work.

Chapter 2

Preliminaries

In this chapter, the basic terminology, methods and fundamental definitions of the time series used in this thesis are presented. Recently a large number of new methods and approaches have originated in the analysis of both economic and financial time series. Conditions of their application have changed with the advent of tools for analysis and also with the increasing extent of the analyzed time series.

The theoretical knowledge in this chapter is based on [16], [19], [36] and [55], where readers can find more theoretical concepts on the issue of time series. The reader is assumed to have a basic knowledge of these topics. The basic principles are briefly summed up in [58].

2.1 Time Series Analysis

A time series is a set of statistics, usually collected at regular intervals. Time series data occur naturally in many application areas. They are created by chronologically grouped data which can be collected by observing a variable over time. Time series are explored by various mathematical, statistical and economic phenomena. The goal of time-series analysis is to understand the mechanism that generates data and to find a suitable method which will be the best for predicting the future trend of observed variables. Time series analysis can be useful to see how a given economic variable changes over time or how it changes compared to other variables over the same time period.

2.1.1 Economic Time Series

Economic time series can be described as empirical observations of the economy, which are disposed in time to a series of values from the past into the present. These time series are distinguished in terms of the length of the time interval of observed values into three groups and these are:

- *Long-term time series* – observed values are in yearly or longer intervals;
- *Short-term time series* – observed values' interval is shorter than one year. It is usually the time period of months or less;
- *High-frequency time series* – the interval of observed values is shorter than a week, mostly days, hours and shorter sections.

The shape of the economic time series is related to this division.

The high-frequency time series include financial time series which are characterized by very short periods of time – they are observed within days. Financial time series and methods that examine them will be described in a separate section. The difference between these series is mainly based on the length of the monitoring interval, and therefore it requires a non-traditional approach to analysis. According to [16], the modeling of financial time series shows that the assumption of linearity and normality is "gross" and this leads to the logical use of nonlinear models. While the space of linear models is considered closed and well researched, the space of nonlinear models has as yet been inadequately explored. If the linear models are not capable of capturing a particular characteristic of stochastic models, then they are transformed and extended to nonlinear models.

Definition 1 *Time Series. An observed value in the time series is usually called* Y*, and its specific value then* $y_1, y_2, ..., y_t, ..., y_n$ *briefly* y_t *wherein the index* $t = 1, 2, ..., n$ *is the index indicating the appropriate interval or time detection and* n *is a length of the time series. The difference* $n - t$ *for a specific value range is called an age of observation (the latest observation has "zero age").*

The characteristic features of economic time series are trend, seasonality, nonlinearity, conditional heteroscedasticity and common properties of multiple time series, such as the common trend. *Trend* reflects long-term behavioral changes in the mean time series, more specifically, it is the general development of a trend of the examined phenomenon for a long time period. It is the result of factors which act in the same direction within the long term, such as e.g. market conditions in a given area. Trend may have a different character, it may be increasing, decreasing, steep moderate and may vary over time so it can be considered a cycle. Seasonality expresses periodic fluctuations around the trend during the calendar year. Seasonal fluctuations are repeated every year in the same period (the length of the period is one year) and they are due to changing seasons or due to different customs, such as holidays, vacations, etc. If single accidental failures have different variances, then the heteroscedasticity model occurs. Heteroscedasticity-forecast errors (errors forecast grow over time) can be verified using the test criterion. The occurrence of some of the properties depends on the type of time series, e.g. heteroscedasticity occurs in high-frequency time series. For purposes of theoretical models, a time series can be understood as a special type of random process.

The procedure of analysis of time series is its decomposition to systematic components, these components are:

- *Trend Component* T_t Sometimes also abbreviated as a trend. As stated above it captures the long-term changes in behaviors of time series. Thus it is not a short-term change over time series, but we are interested in the development of the time series from a long-term point of view. Mostly it is possible to describe the trend component by one mathematical function only throughout the process of time series.

- *Seasonal Component* S_t Seasonal component describes the periodic changes in the time series to take place within one period of time and they repeat every other period of time. An example is a repetition of a certain development of time series in the individual season of the calendar year. Even though this component regularly repeats in time series, it may change its characteristics during an individual season.

- *Cyclical Component* C_t It describes the long-term fluctuations around the trend. So it captures the long-term decline or growth phase, which is much longer than one time period for the seasonal component. The cyclical component for economic time series is often associated with the alternation of economic cycles. Because it acts in the long term, it is very difficult to be traced and described. Moreover, the character of this component may change over time.

- *Irregular Component* ε_t or *noise* Random components are unsystematic (unlike previous components) and are formed from random fluctuations of time series. This component may include all influences that affect the time series and which can not be systematically captured and described. This component is often referred to as *residual component* as it remains after the identification of the three previously described components.

Time series using an *additive model* can be thought of as:

$$y_t = T_t + S_t + C_t + \varepsilon_t \tag{2.1}$$

While a *multiplicative model* would be:

$$y_t = T_t \times S_t \times C_t \times \varepsilon_t \tag{2.2}$$

An additive model would be used when the variations around the trend do not vary with the level of the time series, whereas a multiplicative model would be appropriate if the trend is proportional to the level of the time series.

2.2 High-Frequency Data

One of the ideal sources of high-frequency data (or high-frequency time series) are financial markets. For processing of these data, we need to put together statistical, mathematical, economic and also informatics methods and algorithms. Statistical methods can predict time series well, but the results are not so stable when there is noise in the time series – such as inaccurate or incomplete data. Market data are highly variable and each time interval (known as tick) a new logical unit of data is generated. The main focus of current research is not only the development of high-quality descriptive systems, but also the ability to produce predictions of future movements of data. Information in this section is mainly taken and updated from [30].

In terms of adaptive rules' generation, real-time event processing is a key part we focus on. Fast and automated data analysis will not yield any advantage if every subsequent step in processing requires human approval. The transforming a business system to react in real time requires not only new technologies, but a new way of thinking and solving of the problem as well.

Applications to high-frequency financial data are most apparent, and are characterized by a set of contemporaneously correlated trade marks, many of them discrete in nature at high or ultra high frequency. In empirical studies on financial market microstructure, the characteristics of the multivariate time varying conditional densities (moments, ranges, quantiles, etc.) are crucial.

2.3 Financial Time Series

Financial time series belong to the high-frequency time series that fall under short-time series, the frequency of monitoring being significantly shorter than one year. The reference element of financial time series is the basic information of the financial markets, which is the price. This may be, for example, share price, the price of the currency, bond price and, according to this information, three types of financial markets are distinguished. In financial markets, debt securities (bonds), equities and funds in different currencies are traded. Thus the basic financial time series are based on prices on public markets, or they characterize the prices and their development. These time series have specific characteristics which are significantly different from traditional time series, due to the microstructure of financial markets. A variety of methods focuses on their investigation. These methods and algorithms combine knowledge from different disciplines, such as Mathematics, Statistics, Computer Science and Economics. Statistical methods can predict the time series well, but the results are not too good if we take into account the characteristics of the data, such as the noise in the input data. Data can be inaccurate and incomplete – they may contain outliers or distorted information. High-frequency data are very variable. Each time unit generates new data and these indicate the development of the series.

It is not essential only to process data, but also to predict their course, and thus to estimate trends in the data and to find patterns in the data [60]. The emphasis is on the processing and the prediction of data in real time. Currently, whole platforms exist for the processing of high-frequency data. More on these platforms is given in Chapter 3.

The basic feature of financial time series is a high frequency of recorded values. These values are most often recorded on a monthly, weekly or daily frequency, but may be recorded, for example on the stock exchange, in hourly or minute intervals. Both systematic factors (i.e. impressed and a cyclical trend component) and unsystematic factors, which result in their high and variable variability, have an impact on the dynamics of such fast frequency data recording.

The basic assumption about the behavior of financial markets is the efficient market hypothesis.

2.3.1 Efficient Market Hypothesis

The efficient market hypothesis [42] is derived from the theory of rational choice, which is based on formal models of social and economic behavior. This theory is based on an assumption that every single agent on the market has its own preference function, indicating the utilization of a combination of goods, and performs rational and effective actions to maximize this utilization. The theory of rational choice provide us with models which do not fully describe reality, but these models help us to make decisions which can be considered as rational in the context of maximization of the good utilization. The efficient market hypothesis assumes that financial markets are informationally efficient. This means that all known information is actually involved in the current market price, thus it is impossible to continually over-perform average market revenues by trading with public information. Three forms of this hypothesis have been defined according to its strength.

- *The weak form* declares that extraordinary returns cannot be earned by using strategies that rely on historical movement of prices. Because of this, the technical analysis and methods based on a serial dependence between market prices cannot be used for a prediction of future prices. This form is referred to as weak, because it supposes that

publicly available fundamental information is not immediately flashed into the current prices, thus it can be used for a prediction of the short-term market movements.

- *The semi-strong form* of this hypothesis declares that continuous extraordinary returns cannot be obtained by using publicly available information, because the price is immediately adjusted according to published information. Since there is not enough time between when the information is published and the market is adapted to realize market transactions, there is also no way of utilizing this information. However, the semi-strong hypothesis allows insider traders to use non-public information to achieve profits before markets can adapt to it.

- *A strong form* of the efficient market hypothesis assumes that all public and private information is actually involved in the market prices, thus it is impossible even for insider traders to continually over-perform average market returns. The strong form explains the existence of a small group of investors that excessively and continually over-perform market returns as a result of the normal distribution of earnings over the huge number of individuals.

The efficient market hypothesis does not provide an explanation of market bubbles, crashes and speculative fluctuations of market prices. These fluctuations are caused by similar behavior of a large number of individuals at the same time. If this kind of mass behavior is rational, then there must be rational well-known fundamental reasons which support it. The history of bubbles and crashes reveals that no such reasons were available.

2.3.2 Skewness

Skewness (skew) defines characteristics revealing asymmetry in numerical series and elements of the set. Its symptoms are especially noticeable in high variability. The rate of skewness caused by imbalance can be analyzed by using the distribution and frequency functions. Completely symmetrical distribution has the size of skewness equal to zero. Large frequency of low numbers and small frequency of high numbers have the consequence that the probability density is higher for smaller numbers and the distribution is skewed to the right, otherwise it is skewed to the left [23]. The examples of skewness are shown in Figure 2.1.

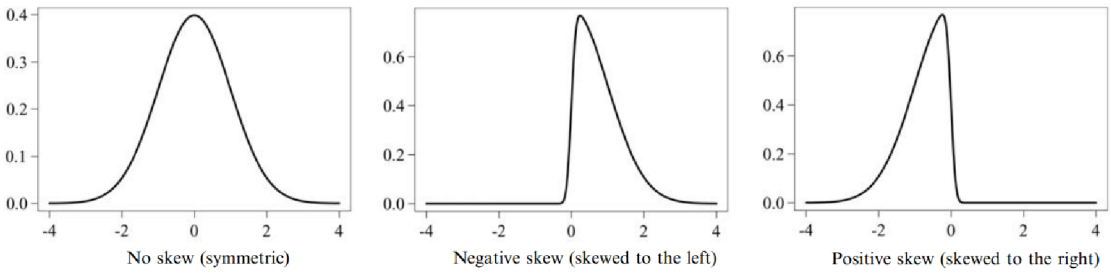


Figure 2.1: Examples of skewness – adapted from [23].

The formula for calculating the skewness, as in the following subsection about kurtosis, is derived from a normalized random variable, which is given in the form of [23]:

$$t = \frac{x_i - \bar{x}}{s}. \tag{2.3}$$

Characteristics of skewness belong among standard moments and are called the third standardized moment for the value. The formula for the general calculation of the standard moment is as follows:

$$\mu_k(t) = \frac{\mu_k(x)}{s^k} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s^k}\right)^k \tag{2.4}$$

Skew helps to better characterize the number series, whose size, mean, median and variance, however, may be identical. From the general formula, by substituting for the third standardized moment, we get the formal registration of calculating the skewness:

$$\gamma_1 = \mu_3(t) = \frac{\mu_3(x)}{s^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{\frac{3}{2}}}. \tag{2.5}$$

2.3.3 Kurtosis

Kurtosis describes eccentricity during the distribution function. When kurtosis is small, it means that it is in the middle course of the function of low distribution of continuous or discrete values. For large values of kurtosis, the probability density functions reach the center of the peak. Characteristics of skewness is the fourth standardized moment. The example of kurtosis is shown in Figure 2.2.

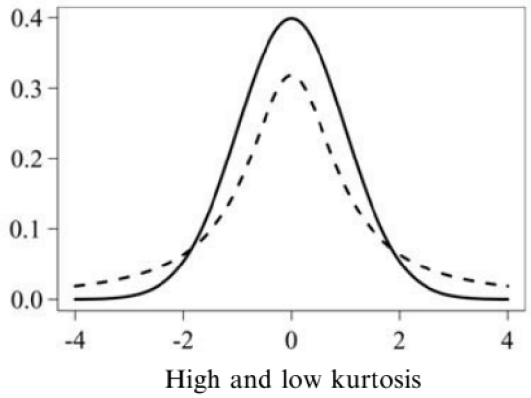


Figure 2.2: Example of kurtosis – adapted from [23].

In order to compare the kurtosis readable, calculating the kurtosis is shifted to the beginning [24]:

$$\gamma_2 = \mu_4(t) - 3 = \frac{\mu_4(x)}{s^4} - 3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4} - 3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^2} - 3. \tag{2.6}$$

Definition 2 *Assumption of normality. Logarithms revenues are normally distributed with constant mean value μ and constant variance σ_r^2. This distribution is characterized by being symmetrical.*

Skewness of this distribution is given by:

$$SK_r = E\left[\frac{(rt_t - \mu)^3}{\sigma_r^3}\right] \tag{2.7}$$

Skewness is equal to 0.
Kurtosis is defined as:

$$K_r = E\left[\frac{(rt_t - \mu)^4}{\sigma_r^4}\right] \tag{2.8}$$

Kurtosis is equal to 3.

Definition 3 *Assumption of linearity. When constructing econometric models, models are often based on theoretical economic models that are in exponential form. Linearization is achieved by logarithm – logarithm-transformed time series must enter into the linear model.*

Assuming the model of the trend is in exponential form:

$$X_t = \sigma\delta^t\varepsilon_t, \tag{2.9}$$

where $t = 1, 2, ..., T$. After that, there is the possibility of using logarithmic transformation to its linearization:

$$\ln X_t = \ln\sigma + \ln\delta.t + \ln\varepsilon^t. \tag{2.10}$$

$\ln\sigma$ parameter characterizes the addition of the series $\ln X_t$ by the change of time t by one unit.

Another feature of the financial market is non-synchronous trading, which is due to the fact that it is not traded on all days of the week and that the new values are not generated at equal time intervals. Liquidity of the data is guaranteed by traders who give orders to buy or sell.

Most linear models for the description of the time series with a stochastic concept are based on the Box-Jenkins methodology. Empirically it has been found that the high-frequency time series are characterized by time-changing variability. This is referred to as changeable volatility, which leads to serious problems when using conventional linear (S)AR(I)MA models.

Linear processes method include:

- ARMA (mixed process),
- ARIMA (integrated mixed process),
- SAR (seasonal autoregressive process) and
- SARIMA (multiplicative seasonal process).

It has been found that the variability may be related to the level and strength of autocorrelation in time series. Characteristic features of such analyzed time series therefore can not be fully captured by a linear model which assumes only one type of dependence – correlation dependence. Nonlinear models are based on a series of nonlinear functions equally distributed in independent random variables; they expect a more general form of dependency than just correlation. Change in volatility (as well as autocorrelation) of the time series can be understood as a change in behavior of the time series. This change can be caused by different factors – deterministic and non-systematic and unpredictable [16].

The theoretical goal of this thesis is to design autonomous algorithmic models based on the principle of statistical arbitrage, offset fair value, correlated time series and the use of multivariate variables and characteristics of the distribution of interim data. These models will be applicable to real-time high-frequency data. The following Section 2.5 explains these principles. The text in this section is based on [27], [29] and [38].

2.3.4 Homogenous Variables in Time Series

In homogeneous time series, the variables such as price, spread, price changes and volatility data rate are examined. With these variables we are able to better describe the dynamics of the market.

Price. The course price of the currency pair at any time is the most watched and most important variable. In addition to the bid price P_{bid} and ask price P_{ask}, the cost of the transaction is defined. Another significant value is the median rate price. This is crucial for the interpretation of current exchange rate fluctuations. The value better approximates the actual price of the currency pair and is formulated as:

$$x(t_j) = \sqrt{P_{bid}(t_j) * P_{ask}(t_j)}. \tag{2.11}$$

The Market Spread. The market spread is also known as the *Bid–ask spread.*The difference between a bid price and its corresponding offer price is known as the spread. It is also known as the market width. The theoretical price in the middle of a spread is known as the midmarket price. The spread of a market is often used as an indicator of its liquidity. The key here is that higher bids are good for sellers and lower offers are better for buyers, so tighter spreads indicate a good deal for both sides. When considering these two hypothetical markets for the same security:

$$5000|1.00 \times 1.13|5000$$

$$5000|1.06 \times 1.07|5000$$

The midprice of both of these markets is \$1.065([1.00 + 1.13]/2). A buyer of 5000 securities in the first market will pay \$5650 whereas in the second market he or she would pay only \$5350. Likewise, a seller of 5000 securities receives \$5000 in the first market but \$5300 in the second. The tighter market is clearly better for both buyer and seller.

Volatility. Volatility is the degree of variation of a trading price series over time, as measured by the standard deviation of returns. The number of repetitions calculating the price change is due to the overall size sample. When using the coefficient $p = 2$ the calculation of volatility is similar to calculating the standard deviations. All price changes are in absolute value, so the result of volatility is also positive.

$$v(t_i) = v(\Delta t, n, p, t_i) = \left[\frac{1}{n} \sum_{j=1}^{n} |r(\Delta t; t_{i-n+j})|^p \right]^{\frac{1}{p}}. \tag{2.12}$$

Volatility is used as an indicator of financial market movements. In markets with low volatility, the rate is almost unchanged. When using a higher rate than the $p > 1$ the calculation is more sensitive to larger changes. The volatility is calculated in the same units as the data source $x(t_j)$. Before the start of the calculation, it needs to be considered at what time intervals the volatility will be calculated and what the total time period will be $\Delta t_{scale} = n\Delta t$. For financial markets, it is typical that the volatility is calculated on the total time interval $\Delta t = 1$hour, while other markets have a normal value of one year. This finding points to the large liquidity of market.

Data Frequency Data that are received from the financial exchange market are not homogeneous. In order to measure frequency, the counting function N must be defined.

This function calculates the number of inhomogeneous data within a specified time interval. The calculation is then trivial.

$$f(t_i) = f(\Delta t; t_i) = \frac{1}{\Delta t} N\{x(t_j) | t_i - \Delta t < t_j \leq t_i\} \tag{2.13}$$

The time interval between quotations currency pair rate is calculated as the inverse value of the frequency $f^{-1}(t_i)$.

2.3.5 Charts

Graphs are the most common and clearest form of representation of information on developments in the currency market. The *line chart* may be placed among charts with the least meaningful value. It shows the progress of prices at different time intervals. According to this chart, we may determine in which direction the market or a particular currency pair is moving. The upward trend rates are characterized by a regular addition to the foregoing quotation currency pair. The opposite trend to the upward trend is the downward trend. In a situation where the development rate for the selected timeframe is declining and also growing, we mention a trend moving sideways. The problem arises in determining the boundaries of the end of the trend. The exchange rate of the currency pair contains waves and peaks, which illustrate the current instability of the market, but this does not mean the end of the trend. The easiest way is to introduce variable deviation to protect from accidental loss. Another method is through the analysis of some of the theories, such as the Dow Theory, or the use of technical indicators, e.g. the statistical characteristics [53]. If there is an upward trend, traders apply the strategy to buy a currency pair; after the trend reaches the end, traders sell. A downward trend is the signal to sell and repurchase upon return to the upward trend.

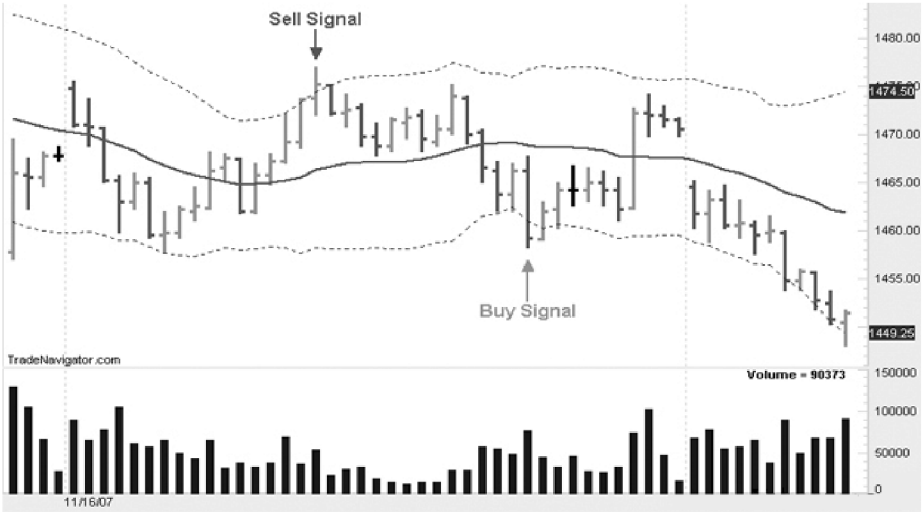


Figure 2.3: An example of bar chart – adapted from [11].

A better inherent value is represented by the *bar (pin) chart*. Each line shows the selected time period – the second, minute, hour, day. The comma is characterized by four prices: Open, High, Low and Close. Figure 2.3 is an example of a bar graph. At the beginning of each section, we take the current course and plot it in a graph as the opening

price. The last price of the currency pair in a given block is shown in the graph closing price.

The *candlestick chart* has practically the same level of information as the bar graph. A candlestick chart is a style of financial chart used to describe price movements of a security, derivative, or currency. Each "candlestick" typically shows one day. It is like a combination of the line chart and a bar chart: each bar represents all four important pieces of information for that day: the open, the close, the high and the low. Candlestick charts are most often used in technical analysis of equity and currency price patterns. They appear superficially similar to box plots, but are unrelated. The description of the single candlestick chart can be found in Figure 2.4.

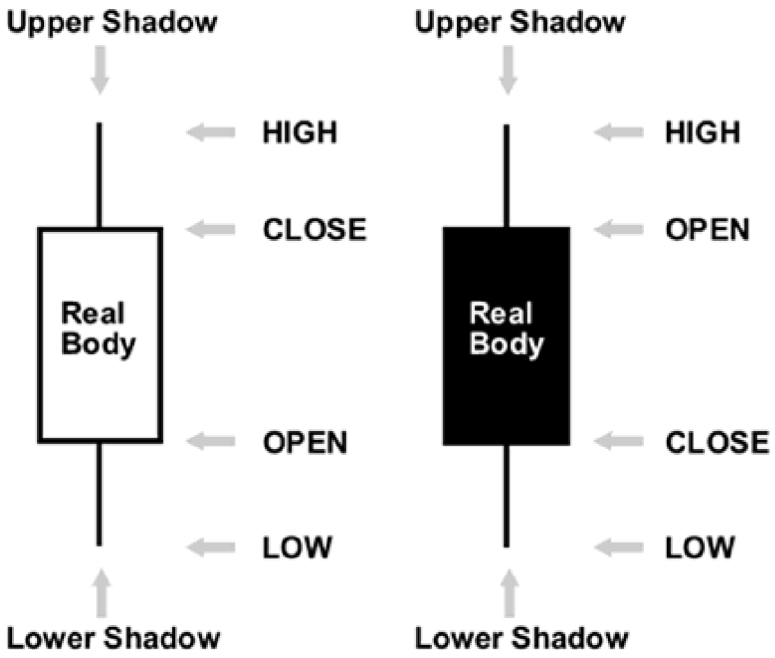


Figure 2.4: Scheme of a single candlestick chart. The Low and High caps are usually not present, but may be added to ease reading [11].

Candlesticks are usually composed of the body (black or white), and an upper and a lower shadow (wick): the area between the OPEN and the CLOSE is called the real body. Price excursions above and below the real body are called shadows. The wick illustrates the highest and lowest traded prices of a security during the time interval represented. The body illustrates the opening and closing trades.

If the security closed higher than it opened, the body is white or unfilled, with the opening price at the bottom of the body and the closing price at the top. If the security closed lower than it opened, the body is black, with the opening price at the top and the closing price at the bottom. A candlestick need not have either a body or a wick. Figure 2.5 shows an example of a candlestick chart.

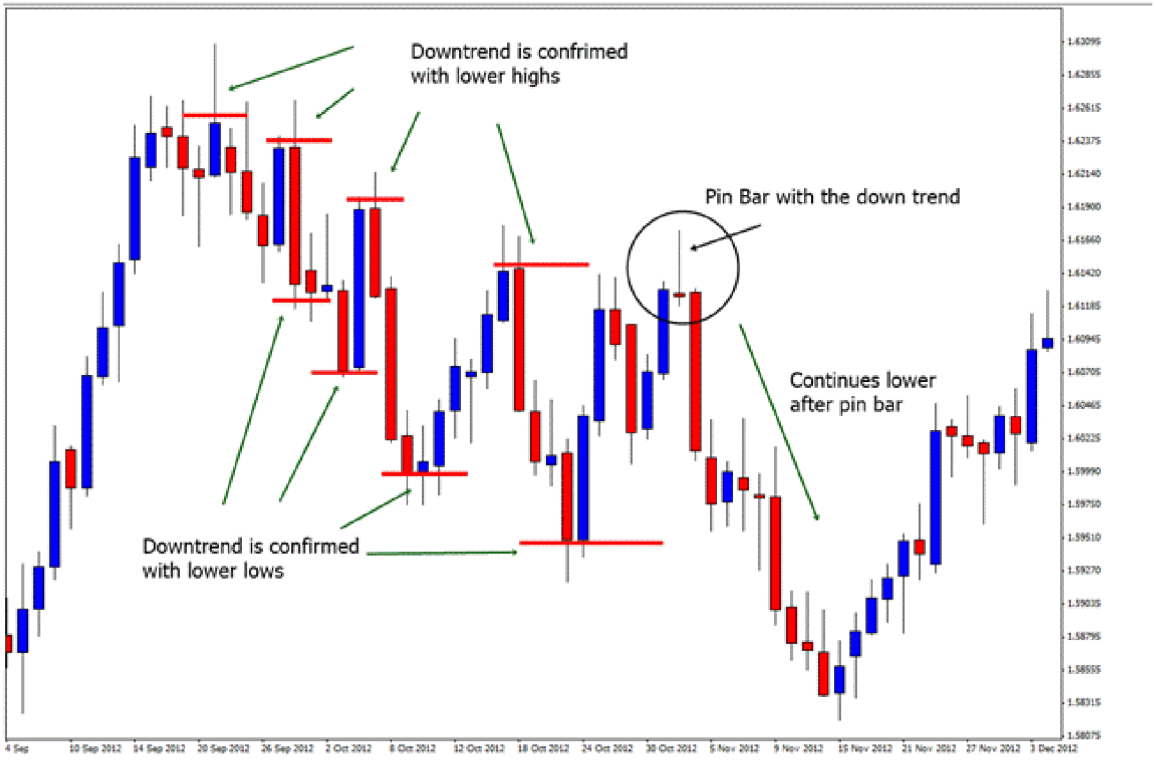


Figure 2.5: An example of a candlestick chart – adapted from [11].

2.4 Moving Average

The trend-following indicators measure the direction and strength of a market trend. The market is trending if the corresponding prices have been continuously rising or falling and if there is a strong probability that this will also be happening in the near future. Trending markets typically involve big fractal efficiency ratios. Concisely, the basic principle of the trend-following strategy is to identify the trend, to open a position in a proper direction and to close it with a profit when the trend matures. We open positions according to the signals provided by one of the trend-following indicators and close them when the opposite indicator occurs, or when a money management rule forces us to do that.

Simple Moving Average is a line which represents the smoothed movement of underlying prices. Points of the moving average line are computed for each bar separately as the average of previous bars:

$$MA_n(i) = \frac{1}{n}\sum_{j=0}^{n-1} c_{i-j}, \tag{2.14}$$

where $MA_n(i)$ is the i-th value of the moving average with the period of n, and c_i is the closing price of the i-th bar.

Exponential Moving Average reflects the short-term memory characteristic of a crowd. The exponential moving average weights closing prices of a chart by numbers from the exponential sequence in order to give more weight to recent prices:

$$EMA_n(i) = \frac{\sum_{j=0}^{n-1} (1-\alpha)^j \cdot c_{i-j}}{\sum_{j=0}^{n-1} (1-\alpha)^j}, \tag{2.15}$$

where α is a small number within the interval $(0; 1)$.

Moving Average Convergence / Divergence, abbreviated as MACD, involves two lines that correspond to exponential moving averages with different lengths. For example, the shorter average can be computed over the last 12 bars, and the longer one over the last 26 bars. These averages correspond to trends with periods of 12 and 26 bars. The turnover of the shorter trend is indicated by the crossing of two EMA lines and the MACD indicator is then computed as a difference between these averages:

$$MACD_{n,m} = EMA_n(i) - EMA_m(i). \tag{2.16}$$

The MACD indicator also involves the MACD signal line which is computed as an exponential moving average of the MACD indicator:

$$MACDsignal_{(m,n,o)}(i) = EMA_o(i) \quad \text{of} \quad [MACD_{m,n}(j)]_{j=0}^{o-1}, \tag{2.17}$$

where $[x_j]_{j=1}^{n-1}$ is a sequence of numbers $x_0, x_1, \ldots, x_{n-1}$.

The relation between the MACD and MACD signal line is represented by the MACD histogram, which is computed as a difference between these two lines:

$$MACDhistogram_{(m,n,o)}(i) = MACD_{m,n}(i) - MACDsignal_{m,n,o}(i) \tag{2.18}$$

The MACD histogram is an ultimate oscillator which indicates the turnover of a trend. A new uptrend starts when the MACD histogram crosses the zero level from the bottom. Similarly, a new downtrend is detected when the histogram crosses the zero level from the top. This concept is illustrated in Figure 2.6.

One problem related to indicators is a fact that they are sensitive to their parameters. This is true also for the MACD as shown in Figure 2.6 – the MACD signaling system is illustrated for a set of parameters (12, 26, 9), but different parameters would produce completely different signals. Because of this, many robust trend-following systems combine several moving averages and they are then known as *Moving Ribbons*.

Section about moving averages based on [42].

2.5 Statistical Arbitrage

Statistical arbitrage is ranked as a business strategy, more specifically it is an investment strategy. This strategy involves pairs trading. It uses statistical measures to detect mispricing between two assets, based on the expected value of these assets. We assume that if the course of these two titles developed historically in the same way, then their mutual deviation will only be temporary. The higher the data frequency, the more arbitrage opportunities appear.

Mathematically, the steps involved in the development of statistical arbitrage (or simply stat-arb) trading signals are based on a relationship between price levels or other variables characterizing any two securities. A relationship based on price levels $S_{i,t}$ and $S_{j,t}$ for any two securities i and j can be arrived at through the following procedure:

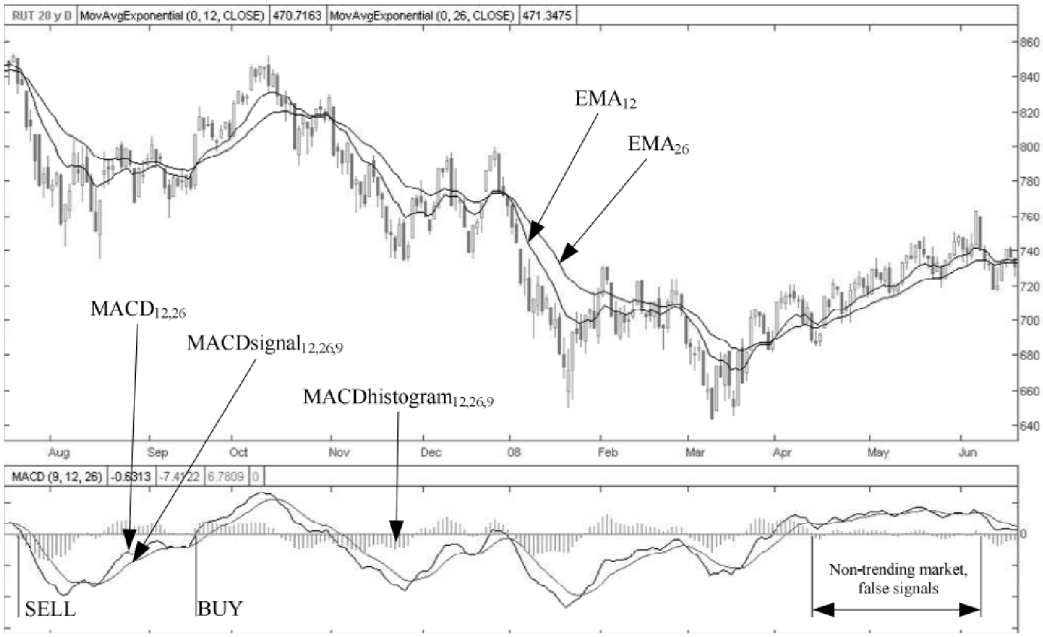


Figure 2.6: The example of the MACD indicator derived from prices of the Russell 2000 Index between the years 2007 and 2008. Adapted from [42].

1. Identify the universe of liquid securities – that is, securities that trade at least once within the desired trading frequency unit. For example, for hourly trading frequency, choose securities that trade at least once every hour.

2. Measure the difference between prices of every two securities, i and j, identified in step (1) across time t:

$$\Delta S_{ij,t} = S_{i,t} - S_{j,t}, t \in [1, T] \tag{2.19}$$

where T is a sufficiently large number of daily observations. According to the Central Limit Theorem (CLT) of statistics, 30 observations at selected trading frequency constitute the bare minimum. The intra-day data have high seasonality – i.e. persistent relationships can be observed at specific hours of the day. Thus, a larger T of at least 30 daily observations is recommended. For robust inferences, a T of 500 daily observations (about two years) is desirable.

3. For each pair of securities, select those with the most stable relationship – security pairs that move together. To do this, perform a simple minimization of the historical differences in returns between every two liquid securities (defined by Franke, Gatev, Goetzmann and Rouwenhorst (2006) [29]):

$$\min_{i,j} \sum_{t-1}^{T} (\Delta S_{ij,t})^2 \tag{2.20}$$

The stability of the relationship can also be assessed using cointegration and other statistical techniques. Next, for each security i, select the security j with the minimum sum of squares obtained in Equation 2.20.

4. Estimate the basic distributional properties of the difference as follows. Mean or average of the difference:

$$E[\Delta S_t] = \frac{1}{T}\sum_{t-1}^{T} \Delta S_t \tag{2.21}$$

Standard deviation:

$$\sigma[\Delta S_t] = \frac{1}{T-1}\sum_{t=1}^{T} (\Delta S_t - E[\Delta S_t])^2 \tag{2.22}$$

5. Monitor and act upon differences in security prices: At a particular time τ, if

$$\Delta S_\tau = S_{i,\tau} - S_{j,\tau} > E[S_\tau] + 2\sigma[\Delta S_\tau] \tag{2.23}$$

sell security i and buy security j. On the other hand, if

$$\Delta S_\tau = S_{i,\tau} - S_{j,\tau} < E[S_\tau] - 2\sigma[\Delta S_\tau] \tag{2.24}$$

buy security i and sell security j.

6. Once the gap in security prices reverses to achieve a desirable gain, close out the positions. If the prices move against the predicted direction, activate stop loss.

This section was based on [14], [29] and [38].

2.6 Fair Value

According to [27] the fair value is *"The calculated price of a given security, typically an option, such that neither counterparty to a trade at that price would experience an economic gain or loss. It is also known as fair market value."*. The impact of changes in fair value is recognized as profit or loss in the period they occur.

According to International Financial Reporting Standards (IFRS) [6], fair value is a market-based measurement and the entity's intention to hold an asset or to settle or otherwise fulfil a liability is not relevant when measuring fair value. When measuring fair value, we use assumptions that market participants would use when pricing the asset or liability under current market conditions.

Characteristics of a particular asset or liability that a market participant would take into account when pricing the item at the measurement date, include: age, condition and location of the asset restrictions on the sale or use, risk characteristics, cost of and return on capital or individually perceived utility.

Fair value is measured by using the price in the principal market for the asset or liability or, in the absence of a principal market, the most advantageous market for the asset or liability.

2.7 Time Series Correlation

In this section, high-frequency time series (or high-frequency data – HFD) and their correlation are introduced. Financial markets are the source of discrete high-frequency data.

The original form of market prices is *tick-by-tick* data. Each tick is one logical unit of information. According to spacing in time, two types of data are distinguished – homogeneous (regularly spaced in time) and inhomogeneous (irregularly spaced in time) [22].

Data typically arrive as a random sequence of time points – the more activity on the market, the denser the data. We study and do research on these data to understand the markets and to predict the behavior of data. Tick-by-tick data allow the market microstructure to be studied and to enable a decision on what type of rules are the most appropriate for the markets to function efficiently. There is need to find new ways of defining the analysis of the data because of its volume: interpolation methods, data cleaning, etc. It is important to develop statistical methods with minimal assumptions of the underlying process.

2.7.1 Correlation of High-Frequency Data

Correlation of the relative measure of mutual dependence in the development of two time series x_t, y_t is given by following relation:

$$s_{xy} = \frac{\sum_{t=1}^{n}(x_t - \overline{x}).(y_t - \overline{y})}{s_x.s_y} \in < -1; 1 > . \tag{2.25}$$

Correlation values approaching the limit value -1 mean that the two time series have completely opposite directions in their time development. Values close to 1 reveal that the time series x and y evolve almost identically in terms of the same direction and have the same relative pace in the mutual development.

Correlation between returns of different financial assets plays an important role in fields such as risk management [29]. One of the known problems concerning correlation is that correlations between financial time series data vary over time.

2.8 Multivariate Random Values

In mathematics, probability and statistics, a multivariate random variable or random vector is a list of variables with unknown values, either because the value has not yet occurred or because there is not good knowledge of its value. The individual variables in a random vector are grouped together because there may be correlations among them – often they represent different properties of an individual statistical unit (e.g. a particular person, event, etc.). Normally each element of a random vector is a real number [38].

Random vectors are often used as the underlying implementation of various types of aggregate random variables, e.g. a random matrix, random tree, random sequence, random prices, etc.

More formally, a multivariate random variable is a column vector $X = (X_1, ..., X_n)^T$ (or its transpose, which is a row vector), whose components are scalar-valued random variables on the same probability space $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{F}$ is the sigma-algebra (the collection of all events), and P is the probability measure (a function returning each event's probability) [18].

Chapter 3

Complex Event Processing for High-Frequency Data

Complex event processing (CEP) is an emerging technology that generates actionable knowledge from distributed message-based systems, data streams and historical data in real time or near real time. There are several CEP engines, but only a few are capable of integrating data from multiple sources and working with high event volumes. Environmental measurements' processing and making widespread use of it is a big data problem. The emphasis on real-time data processing is becoming an essential requirement. Though CEP provides mechanisms for computing of high volume of events, it does not define any methodologies, models and standards, which would establish any architecture model as a mature software architecture [44].

According to DAvid Luckham [40], we can abstract many different levels of CEP. For example, if we consider trading and financial markets, at the lowest level there will be a stock trader responsible for executing trades. A trade might consist of several executed bids, offers, payments and other financial transfers. A complex event, indicating how much profit or loss the trader generated during a specific time interval, has predictive power which can be used in subsequent decisions [22], [40].

Complex event processing offers its users a method of automating the detection of anomalies or other detectable and exploitable phenomena. It is too laborious for the auditor to correlate all the trades carried out by all the traders to detect all the various errors they might have made. On the other hand, a CEP system, if properly configured, is able to react more rapidly than a human. The adaption of the rules can speed up the process by an automatic set up of process on the runtime as a response to a specific change in context.

3.1 CEP Characteristics

The process of analyzing events and finding situations of interest is known as 4D model (Detect – Derive – Decide –Do). CEP detects and derives information, so we can become aware of a situation and react to it immediately.

The 4D model is decribed by following four parts of event processing:

- *Detect* – This is the capturing the event that comes from input event sources. This might be a sensor reading or a request to do some action.

- *Derive* – This is the act of correlating of an event with other events or other derived understanding of the context of interest.
- *Decide* – The decide is determining if and what to do because of the awareness of the situation.
- *Do* – This part represents doing the activity that was decided to be done.

The use of 4D model goes beyond the capabilities CEP, it also uses the approach of service oriented architecture or reative systems. The "Detect" and "Derive" sections are the responsibility of CEP.

The "Decide" part might be handled by decision support tools or rules engines, as their strength lies in decision tables and fact based analysis. The "Do" is sometimes handled by business process or workflow tools. CEP plays a large part in this because we can take various events, combine/aggregate them according to defined patterns and derive other events.

The traditional approach for detecting anomalies on the stock market has been statistical analysis after the trades have been completed. For rapid reactions to fraud, real-time monitoring is a necessity and it can be used as a complementary method. The approach based on streaming data is proved to be several orders of magnitude faster than the traditional approach based on a relational database and single issue queries [12].

There currently exists a number of complex event-processing platforms. In general, these platforms are a set of tools that support the preprocessing, processing and prediction of complex events. These platforms are designed for processing of data from multiple different sources and primarily focus on processing of moving data streams in real time. These data are processed on several levels of abstraction according to the required level of interference. The output of the process is pattern recognition, mining of trends and patterns in data and so predicting the flow of subsequent input data.

Beside these platforms, there are other tools supporting complex event processing, such as frameworks, libraries, modules, etc. The current CEP tools do not solve identical problems, so it depends on the purpose for which the user wants to utilize these tools. Tools for CEP can be further divided according to the data characteristics.

We focused on tools which are designed for processing of high-frequency data. As stated above (2.3), these simultaneous events occur in very small time intervals coming from multiple sources and in high volume. These data can be labeled as high-frequency time series. They need to be described by nonlinear models. High-frequency data can be understood as defined by a time series of consecutive events. Examples of such data may be data telecommunications, pharmaceutical, energy or financial data. Data processing using complex platforms is based on a layered model, as is the case e.g. in a layered model Internet – ISO / OSI. CEP has been used for various purposes, such as fraud detection, algorithmic trading, supply chain monitoring, network management, traffic monitoring, call monitoring, etc. CEP is often used in combination with service-oriented architectures (SOA). Information about CEP in this section is based mainly on [41].

For the description of data different formats are used. E.g. IFRS define the format of financial data, energetic data might be decribed by using CIM (Common Information Model) format (previously UCTE).

3.2 Events

There are two parallel definitions for the concept of an "event". Firstly, it can mean anything that happens, or is happening, e.g. a financial trade is carried out. Secondly, it may be seen as the object acting as the manifestation of something that happens, e.g. a purchase order is sent [40]. Because of the nature of the CEP, we will use a representation-based definition of 'event'. A corresponding *event object* carries general metadata (i.e. event ID, timestamp) and event-specific information, e.g. a sensor ID and some measured data.

In complex event processing, multiple rules are applied to the events that flow through the system. These rules are applied in Event Processing Agents (EPA), which are the fundamental building blocks of CEP. EPAs monitor the patterns in event flows and react according to the defined function. They take events as input and produce new events as output according to the given set of rules. Luckham [40] classfies agents into three groups:

- *Input filters* – Filters are event patterns that remove irrelevant events from the streams. Only relevant events are passed further to maps and constraints.
- *Maps* – Maps are used to create higher level complex events by aggregating multiple lower level events. These aggregations specify event hierarchies.
- *Constraints* – Constraints can detect the presence or absence of an event or a complex event in a stream. They create notification events, when the constraint is violated (broken).

In [28], Etzion and Niblett define nine different EPA types: *filter, pattern detect, transform, aggregate, split, compose, translate, enrich* and *project*. The most important types are filter, transformation and pattern detect. Transformation is an abstract supertype of translate, aggregate, split and compose, and never used as such alone. Translation means directly mapping one event to another event. This part was based on [12].

An event is associated with a source that acts as a source generator. These generators can be:

- *structure operation*, events related to operations that operate on data structures;
- *behavior invocation,* events related to invocation of user-defined operations;
- *transaction,* events related to database transactions;
- *abstract* or *user-defined,* explicit signaling of events in application programs;
- *exception,* not able to reach data in a database system;
- *clock,* time events;
- *external,* events raised outside a database system.

In Figure 3.1 a schema of a complex event architecture is presented. Processing of events is divided into several levels which conform to the desired level of inference. At the lowest level, the event preprocessing runs – during this phase we clean the input data stream to produce comprehensible data. On the next level, the events that were detected in input data are refined and subsequently initial decisions and correlations are made. The main challenge is to find the relevant data. Thereafter, situation refinement and impact

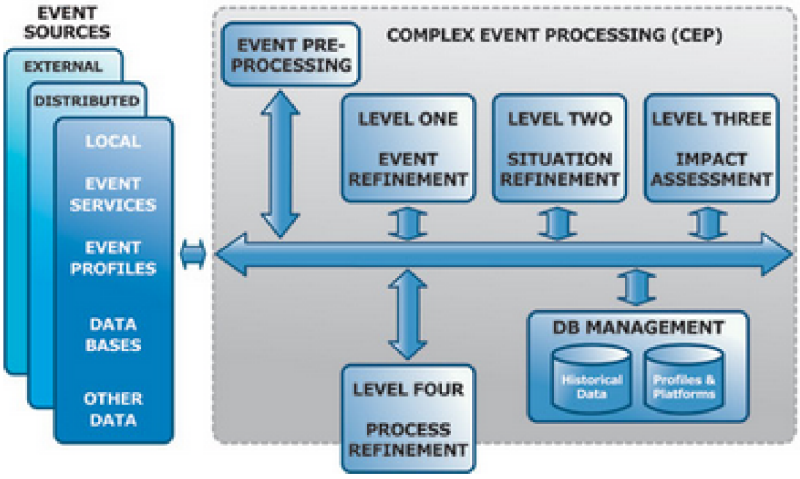


Figure 3.1: Complex event processing reference architecture – adapted from [8].

assessment follow. At the level of impact assessment, we may predict the intentions of the subject or estimate potential opportunities or threats. Finally, process refinement is carried out. Information based on [60].

All the results of event processing and operational visualization at all levels may be summed up in a humanly readable format via user interface.

Most current CEP platforms' solutions fall into one of these two categories:

1. Aggregation-oriented CEP, or
2. Detection-oriented CEP.

The first approach uses real-time processing of event data which enter the system. As an example, we might take an algorithm, performing some calculations within a moving window of a given size. On the other hand, there is the detection-oriented solution which focuses on examination of data and detection of patterns or recurring behavior. Many applications use a combination of both approaches.

3.3 Event Driven Architecture

Current software architectures do not target event-based systems, because they are mostly based on a process-oriented control flow, which is not sufficient for event-driven systems. In recent years, Event-Driven Architecture (EDA) has been proposed as a new general processing model for event streams [40]. The key concept is to use CEP as a process model for event-driven decision support. Event streams (streams of ticks) emitted on a market contain a high volume of events, which must be transformed, classified, aggregated and evaluated to initiate appropriate domain actions. Although CEP provides mechanisms for computing a high volume of events, it does not define any methodologies, models and standards which would establish EDA as a mature software architecture [26], [44].

Unfortunately, event-driven architectures have not yet the maturity of well-established software architectures: there is still a lack of methodologies, models and standards [26].

EDA provides an architectural concept which deals with the processing of continuous events' streams. The control flow of EDA is based on the processing of multiple types of events from different sources.

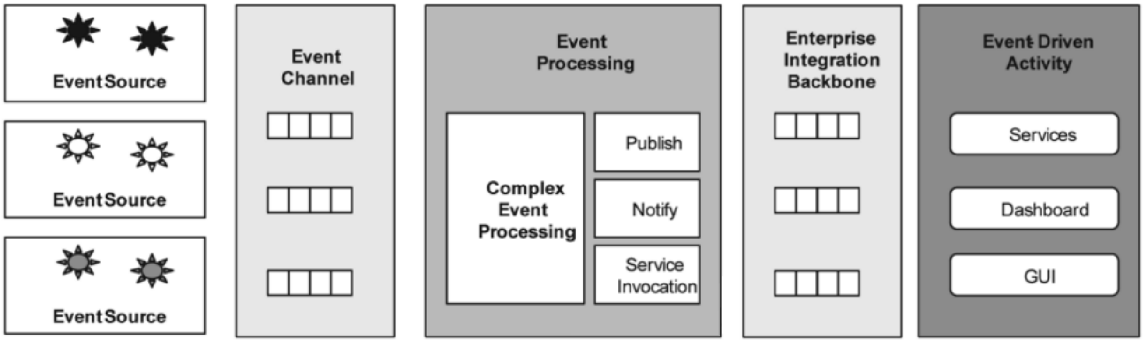


Figure 3.2: Flow of events in event-driven architecture – adapted from [44].

Event flow in EDA, shown in Figure 3.2, can be divided into several components:

- *Event source* – Events are generated by an event source. For market data, the sources of events are the transactions made by traders. Each event must be described in a standard format, e.g. XML, CSV, etc. The syntax of event description should be well-defined by using a meta language.
- *Event channel* – An event channel provides the infrastructure for the events to the event-processing components. A message-oriented middleware (MOM, i.e. ActiveMQ) serves for the sending of events in the form of messages. MOM contains several different message queues and each queue is dedicated to a certain event type. The communication then passes as follows – messages are forwarded to those components which have subscribed for the corresponding event type.
- *Event Processing* – At this level, the analysis of continuously arriving streams of events runs according to the CEP concept, as was described above in this chapter. Alternatively, events can be published to other components or initiate a notification for a human operator.
- *Enterprise Integration Backbone* – The Enterprise Integration Backbone (EIB) provides the infrastructure to connect to the event-processing component with the enterprise backend system, e.g. information system.
- *Event-Driven Activities* – The real handling of the event is not provided by CEP, but by the operational business applications and backend systems. This component of EDA provides activities which implement the domain-specific event handling and also the dashboards or graphical user interfaces for visualizing of events.

Source data can be refined as three main blocks of events:

- Event Metadata – automation of event processing requires a formalism based on metadata. An event model should provide a complete understanding of the different event types, its properties, constraints and dependencies. The event model is the base for the subsequent event processing.

- Event Processing Rules – can define correlations between events for detecting events' patterns and determining corresponding actions. The rules consist of two different parts: event patterns specify a certain situation of events, and event actions are executed when the event pattern is fulfilled, i.e. it matches. New events can be generated within the event action part. CEP relies completely on the event model, where all events which used the event-processing rules must be defined. Event Patterns are based on event types, i.e. they define a sequence of event types that must be matched. If the events must be processed in order, an event sequence path can be defined.
- CEP Patterns – are described in the following section.

3.4 CEP Patterns

Events are elementary units which are processed by CEP. Simple events may be aggregated into more complex events, which create a pattern of events. Patterns of events are usually set up by humans. This creates the risk that the system may include errors caused by the human factor. Another common source of error when creating a pattern of events is noise in the input data. For this reason, CEP platforms contain a comprehensive platform-level data preprocessing filter data using double-checking [40]. CEP analyses continuous streams of incoming events in order to identify the presence of complex sequences of events, so called event patterns. A pattern match signifies a meaningful state of the environment and causes either creating a new complex event or triggering an appropriate action.

Esper, the CEP tool we are using for the events processing, can apply match-recognize patterns in real-time upon arrival of new events in a stream of events. Esper can also match patterns on-demand via the iterator pull-API, if specifying a named window or data window on a stream. Using match recognize patterns are defined in the syntax of regular expressions. More about the Esper and its components is in one of the following sections (Chapter 5, Section 5.3). Once we have historical data, we may simulate the designed solution with dynamically set-up rules on these data. After the run, we will compare the experimental results with real data processing. Pattern detection always functions in some context. The context defines the relevant events impacting the pattern matching. It can be temporally or spatially bounded. Context can also be based on semantics of mutually referenced objects or entities. This context is called a window. According to [12], the patterns are divided into two categories – basic patterns and dimensional patterns.

For example, there are patterns to detect if one instance of each type of the participant set (or none of them) has been seen. These patterns without clearly defined rules can be very ambiguous. Pattern policies allow us to express evaluation, cardinality, repeated type, consumption and order policies. Evaluation policy defines whether we want to evaluate the pattern every time a new event is observed. Cardinality policy determines how many times one event can be part of a matched pattern [12] and [22].

The traditional approach for detecting anomalies in the stock market has been statistical analysis, after all the trades have been completed. The CEP solution has the benefit that analyses according to patterns can be run when needed. CEP is suitable for fast reactions to fraud, when real-time monitoring is a necessity and can be used as a complementary method. For example, we may detect a fraud for payment by credit cards according to a spatial restriction – if we encounter two payments from different places with a very long distance between them, within a short period of time, we can almost be sure that it is a fraud.

CEP systems are often developed bottom-up by first identifying the event information available. However, in [13] and [35] a top-down approach is described. First of all, the key performance indicators and other abstract measures are defined and then we hierarchically proceed down to find the correct low-level events in a changing environment in order to calculate them.

CEP distinguishes several scalability attributes:

- *Events volume*
- *Event processing agents*
- *Producers and consumers*
- *Window size*
- *Computational complexity*
- *Environment*
- *Constants*

For stream analytics, it is a key capability that complex event-processing systems are able to scale out in order to process all incoming events in a timely fashion as required by the application domain. The basics about CEP patterns are also summed up in [59].

3.5 Rules for Real-time Processing

In [50], rules for real-time processing systems are introduced. As we want to design a model for event processing which can react in real time, we will design the model with respect to these requirements. We return to these rules in Section 6.8.5 and evaluate the implemented system with respect to them.

The rules and their brief introduction:

1. *Keep the data moving* – to process messages "in-stream", without any requirement to store them, to perform any operation or sequence of operations in order to achieve low latency. An additional latency problem arises for systems that are passive, meaning that the system requires applications to continuously poll for conditions of interest.

2. *Query using SQL on streams (StreamSQL, CQL)* – a traditional SQL system knows when the computing is complete when it gets to the end of a table, but the streaming data never ends, so the stream-processing engine must be instructed when to finish such an operation and output an answer. The window concept serves this purpose by defining the "scope" of a multimessage operator such as an aggregate or a join. Depending on the choice of window size and slide parameters, windows can be constructed as isolated or overlapping.

3. *Handle stream imperfections (delayed, missing and out-of-order data)* – we need to have built-in mechanisms to provide adaptability against stream "imperfections", including missing and out-of-order data, which are commonly present in real-world data streams.

4. *Generate predictable outcomes* – a stream-processing system must process time-series messages in a predictable manner to ensure that the results of processing are deterministic and repeatable. The ability to produce predictable results is also important from the perspective of fault tolerance and recovery.

5. *Integrate stored and streaming data* – requires the capability to efficiently store, access and modify the state of information, and to combine it with live streaming data. For seamless integration (without modifying the application code), the system should use a uniform language when dealing with either type of data.

6. *Guarantee data safety and availability* – to preserve the integrity of mission-critical information and avoid disruptions in real-time processing, a stream-processing system must use a high-availability solution. We must ensure that the applications are running and available, and the integrity of the data is maintained at all times, despite the failures.

7. *Partition and scale applications automatically* – to have the capability to distribute processing across multiple processors and machines to achieve incremental scalability. Stream-processing systems should also support multi-threaded operation to take advantage of modern multi-processor (or multicore) computer architectures. Ideally, the distribution should be automatic and transparent.

8. *Process and respond instantaneously* – a stream-processing system must have a highly optimized, minimal-overhead execution engine to deliver real-time response to high-volume applications.

3.6 Foreign Exchange Market

The foreign exchange market, abbreviated as FOREX or FX, is the largest and the most liquid financial market in the world with over $5.3 trillion worth of trades carried out every day. Forex is a great source of high-frequency data which can be further processed by CEP. Unlike other markets, FOREX is an over the counter (OTC) market – there is no centralized exchange depository or exchange for FOREX trading. It is a global decentralized marketplace that determines the relative values of different currencies. Instead, these transactions are conducted by several market participants in several locations. It is characterized by low margins and high leverage. FOREX is an interbank market: the core players in the FOREX market are central banks, commercial banks and investment banks. The prices on the foreign exchange market are determined to a large extent by these interbank participants. FOREX trading is a simultaneous buying of one currency and selling another. Currencies are traded through a broker or dealer and are traded in pairs; for example the euro and the U.S. dollar (EUR/USD). When someone trade in the FOREX market, he or she buys or sells in currency pairs. The exchange rate between two currencies constantly changes. An increase in supply or a decrease in demand for a currency can cause the value of that currency to fall. While the established financial institutions use expensive systems to execute trades, e.g. ultra-low latency direct market access software, individual investors only have a few simple tools at their disposal. Affordable software exists and integrates well with brokerage services. It often allows the execution of custom trading algorithms. However, it does not allow the analysis of rich financial data, which is crucial to making informed trading decisions or building trading algorithms. FOREX trades 24 hours a day,

5 and a half days a week. Trading moves across borders and around the globe with the clock.

Currently, a growing segment of the FOREX segment spot transactions goes through automated, electronic order-matching systems, such as Electronic Broker Service (EBS). These markets deliver good high-frequency data with transaction prices and volumes. The bid-ask prices from the OTC FOREX market are called *quoted* prices or simply quotes. One full tick contains the time stamp, a bid and an ask price and the origin of the tick.

A *quote* is always comparing one currency to another and it is read as follows, e.g. the EUR/USD at 1.4022 shows how much one euro (EUR) is worth in U.S. dollars (USD).

A *lot* is the smallest trade size available. Account holders can place trades of different sizes, but they must be increments of 1000 units, like 2000, 3000, 15000, 112000 etc.

A *pip* is the unit in which a profit or loss is counted. Most currency pairs are quoted to four decimal places. The fourth decimal place after the decimal point is typically a "pip". Every point that place in the quote moves is 1 pip of movement. For example, if the EUR/USD rises from 1.4022 to 1.4027, the EUR/USD has risen 5 pips.

The main characteristics of the FOREX market are: lower trading costs, excellent transparency, superior liquidity and very strong market trends.

FOREX spot, futures, options, and exchange-traded funds (or ETFs) belong among the most popular ways of investing or speculating on the FOREX market.

3.6.1 Spot Markets

On the spot market, currencies are traded immediately ("on the spot") at the time of the transaction using the current market price. The characteristics of this market are its simplicity, liquidity, tight spreads, and round-the-clock operations. It is the most original form of trading but it has some disadvantages. The timing is not flexible, traders have to deal with the delivery of the traded asset. The FOREX market is a major example of a market where the spot trading is still strong. But in some cases derivative markets become more important than spot markets.

3.6.2 Futures Markets

Futures markets have a higher liquidity and volume than the spot markets and produce better high-frequency data. Futures are contracts to buy or sell a certain asset at a specified price at a future date. Since futures contracts are standardized and traded through a centralized exchange, the market is very transparent and well-regulated. This means that price and transaction information is readily available. The structure of the futures markets has changed due to the rapid growth of the market volume and shifted to electronic trading.

3.6.3 Options Market

An "option" is a financial instrument that gives the buyer the right or the option, but not the obligation, to buy or sell an asset at a specified price on the option's expiration date. Option prices are very variable. If a trader "sold" an option, then he or she would be obliged to buy or sell an asset at a specific price at the expiration date. Just like futures, options are also traded on an exchange. However, the disadvantage in trading FOREX options is that market hours are limited for certain options and the liquidity is not nearly as great as the futures or spot market.

3.6.4 Exchange-traded Funds

Exchange-traded funds or ETFs are the youngest members of the FOREX markets. An ETF could contain a set of stocks combined with some currencies, allowing the trader to diversify with different assets. These are created by financial institutions and can be traded like stocks through an exchange. Currency ETFs aim to replicate movements of currency in the foreign exchange market by holding currencies either directly or through currency-denominated short-term debt instruments. Also, since ETFs contain stocks, these are subject to trading commissions and other transaction costs. With the growing popularity of ETFs it is relatively easy and inexpensive to trade currency ETFs in order to take advantage of fluctuations between currencies.

Information about FOREX market based on [22]. In the attachments the table with the currencies traded on FOREX (Table B.1) and the chart of the market size can be found(Figure C.1 and Figure C.2).

Chapter 4

Decision-Making Process

This chapter summarizes the decision-making process in complex event-processing platforms. Thereafter, the decision support systems (DSSs) will be described, and a more detailed description of a knowledge-based DSS will be given. This type of DSS will be designed and implemented within the scope of this thesis.

This chapter also gives a brief overview of business rules and how to record and maintain them.

4.1 Decision-Making Process in CEP

Recognizing decision making as one of the most common but significant functions which management has to perform on a daily basis, it is important that organizations pursue to improve the processes and efficiency of the decisions made. The dynamic development of information technologies creates the possibility of using them in modeling a dynamic management process and in support of the decision-making process.

A decision-making process in CEP is implemented as stateful. This means that the decisions are not based merely on the actual data that come to a system, but historical sets of data are also taken into account. Decisions depend on other parameters, such as the context of events, time, etc. CEP deals with relations among events of different situational types and thus can determine assessments and trends in data. For the decision-making of some more complex situation which requires calculation a decision-making engine which communicates with the CEP solution can be used. An existing tool which can generate action as an output based on a given set of data (e.g. FICO Blaze Advisor, www.fico.com) can be used or user solution can be implemented. The result is still set up on the fly without the need of redeployment of the running process. Communication with external solution might be provided via web services – this design is used in Service Oriented Architecture (SOA) approach. Service oriented architecture (SOA) models were created to facilitate the design of enterprise software [59].

SOA addresses the following concerns:

1. many systems need to be integrated to a single interoperable entity and serve as a service for other systems

2. the existing components don't have to be implemented strictly in the same language in order for them to communicate to each other

3. businesses implement new products rapidly. Another source of integration requirements are mergers, which bring new, incompatible systems to the ecosystem. Desinged model must scale to support high volumes of events [12].

The decision-making engine uses predefined rules to identify situations. Figure 4.1 shows a schema of the decision-making process in CEP. This schema is based on the StreamBase CEP model [9].

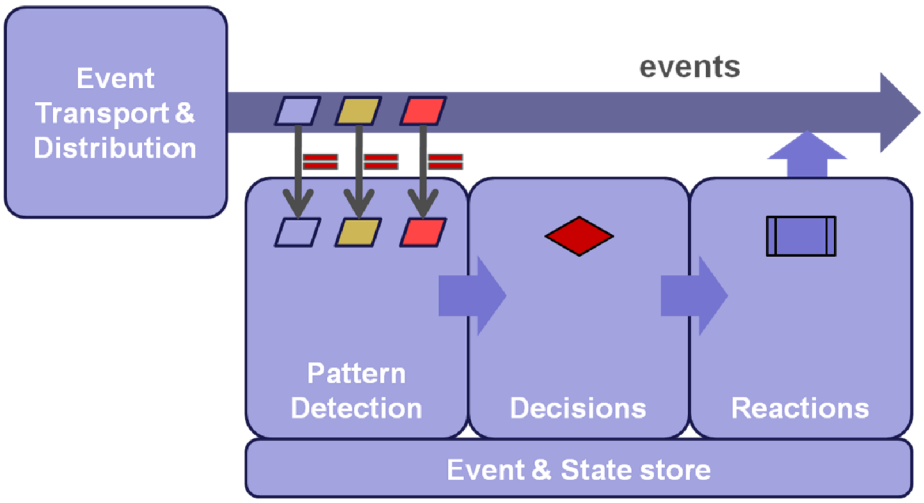


Figure 4.1: Decision making process schema – adapted from [9].

In the first part of this process, the patterns are recognized The detection is followed by the decision-making and the reaction to the detected phenomena and then we make decisions and react to them. In the second step – decisions – we use the set of business rules. This set contains the business rules which affect further processing of events' flows and enables the addition of newly recognized patterns and rules. This should be done automatically in real time when the process is still running. At this point, we focus on the set of rules. In this work we want to formally describe the set of business rules by matrix grammar and the dependencies between the rules will be represented by the matrices of rules. Matrices allow us to model restrictions of the business process. In this step of processing, other tools supporting decision making can be used e.g. decision tables, vocabulary support.

4.2 Decision Support System

Decision Support System (DSS) is a computer-based information system or subsystem that supports complex business or organizational decision-making activities. The organizations are *founded* on decisions, the businesses of organizations are *based* on decisions. Decision making therefore cuts across every segment of an enterprise. For decision making to be successful, the information on which these decisions are based should be reliable and accurate. DSSs serve the management, operations, and planning levels of an organization and provide assistance in decision-making processes. Any computer application that enhances an individual or group's ability to make decisions is considered as a DSS. Decision support systems can either be fully automated, human or a combination of both. The main DSS' goal is improving decision-making efficiency, not automating decisions. The fundamental task for a modern DSS is to assist decision makers in building up and exploring

the implications of their judgements. Due to the high volume of events and their complex dependencies, no predefined workflow can be specified [44]. Workflows are set up according to the characteristics of input data.

4.2.1 Development of DSS

Main requirements for development of DSS:

- Speed – the processing system is required to work in real-time or nearly real-time even though it is overloaded with information and there are distortions of information;
- Fact-Based Decision Making – a complex decision-making environment creates a need for automatic decision support. A well-designed and appropriate computerized decision support system can encourage fact-based decisions;
- Improvements of Decision Quality – along with improvement of system effectiveness.

The methodology can be defined as an organized set of practices and procedures used by developers. Despite many differences in methodologies and terminology, the prescriptions in the Information Systems literature have generally followed three different conceptual paths.

- Design and development of traditional information system based on analysis and design literature.
- Iterative, prototyping, or "quick-hit" approach for designing and developing of DSS'.
- a third approach to building DSS is called end-user development and. The main idea is to let managers develop their own personal DSS.

In general the DSS approaches on design and development is based on personal experiences, case studies and the general IS development literature.

4.3 DSS Architecture

Current software architectures of decision support systems cannot deal efficiently with the processing of continuous event streams. Existing approaches focus on knowledge processing, but do not explicitly target the problems associated with real-time event processing.

Figure 4.2 shows a schema of a decision support system. The main components of a DSS are data, models for data processing and knowledge. Data can be external or internal, including all the information about the processes that need to be covered by the DSS. Models of the data are usually used for accounting and financial analysis, simulation models and for evaluation of plans. Information supporting decisions can also be input into the system by its user through an appropriate user interface.

DSSs are frequently mistakenly confused with decision management systems (DMS). The difference is that a DMS makes decisions without further human interaction. However, this system just makes a decision, it is not responsible for the workflow. According to James Taylor [51], both DSS and DMS apply expertise and judgment, but DSS relies on the user to have experience and apply his or her own judgment. This means that decision support functions better for strategic and management/control decisions where the user is likely to

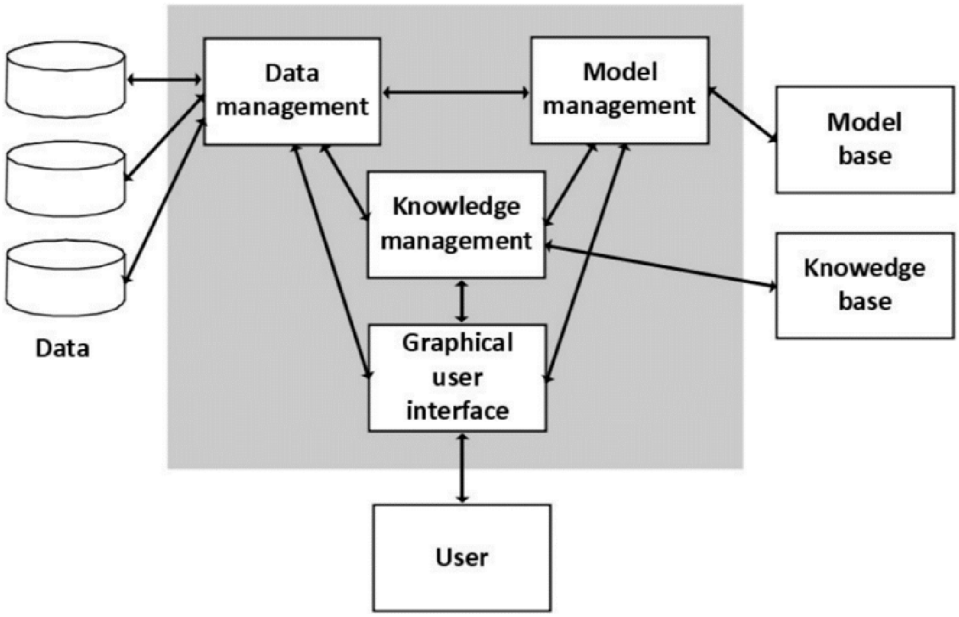


Figure 4.2: Schema of a Decision Support System – based on [32].

have some significant experience. Decision management is a better approach to operational decisions.

DMS uses automated decisions — this means that the decisions are automated for 100% of time and in 100% of cases. Many automated decisions are neither 100% of decisions nor 100% of each decision. Such systems are not pure Decision Management Systems nor are they pure Decision Support Systems – they are a fusion of both systems. Nevertheless they can be very effective [2].

Decision Management Systems are built by focusing on the repeatable, operational decisions that impact individual transactions or customers. Once these decisions are discovered and modeled, decision services are built that embody the organization's preferred decision-making approach in operational software components. The performance of these components, and the impact of this performance on overall organizational performance is tracked, analyzed and fed back in order to improve the effectiveness of decision making. The DSS model component is created by the defining of business rules which can be found within the business process. Information in this section is based on [2], [25], [47] and [51].

4.3.1 Classification of DSS

There are several classifications and taxonomies of DSS applications. Classification divides DSS into five main categories. The current, most common and widespread DSS classification, popular with many authors, such as Power [47] or Turban [56], is as follows:

- *Data-driven DSS*, which is primarily based on the data and their transformation into information. These systems usually analyze a large volume of data; they support decision making by allowing users to extract useful information. Data are collected in data warehouses for this purpose. Online analytical processing and data mining can then be used to process the data.

- *Model-driven DSS*, which puts the main emphasis on the use of simulation and optimization models. Earlier DSS systems were mainly model-based standalone systems.
- *Knowledge-driven DSS*, characterized by the use of knowledge technologies to meet the specific needs of the decision-making process. Usually consists of knowledge about a particular domain.
- *Document-driven DSS*, that assists users to acquire and process unstructured documents and web pages and thus provides complete document retrieval and analysis.
- *Communication-driven* and *group DSS*, which includes all systems built using communication technologies to support collaboration of user groups.
- *Hybrid DSS* – all the above listed categories can be combined to create compound or hybrid systems.

Information about classification of DSS is based on [32] and [47]. We classify newly designed system as a knowledge-based driven DSS. The model part will be described by the business rules.

4.3.2 Complex Event Processing for Decision Support Systems

Complex Event Processing is responsible for processing streams of continuously arriving events, i.e. the operational or process behavior of EDA. The event hierarchy defined in the structural event model corresponds to the sequence of event-processing steps. Event processing is in fact event transformation: raw sensor events are transformed into more abstract and sophisticated application-specific events for initiating appropriate control steps. The subsequent stages of event processing yield the basis for the software architecture. The event transformation steps are processed by corresponding event processing agents (EPA), which are connected to an event-processing network (EPN) [44].

The decision types include scheduled decision problems (routine, repetitive task, well-structured, easy to solve) and unscheduled decision problems (new, unstructured, difficult to resolve). The DSS field participates in this latter type as a computerized system for semi-structured or unstructured decisions. A computer system could be developed to deal with the structured portion of a DSS problem, but the judgment of the decision maker is brought to bear on the unstructured part, hence constituting a human–machine, problem-solving system [47]. In addition, other systems interact with the DSS. Data Mining and Knowledge extract patterns from massive data sets for decision support.

4.4 Business Rules

Business rules should support the decisions of the business, not just describe the technical (fixed) conditions within the system. The recognition of business rules is not carried out fully automatically. There are decisions that cannot be made without human interactions. Nevertheless, on the other hand, there are many situations, business opportunities or threats that can be detected by the use of historical data and known trends in data.

In [57] a method is described for recognizing business rules within an organization. This recognition is split into several parts. In the first phase, business analytics extract the rules stated in sentences of natural language. These sentences are then associated to a specific part of the business process. During the next phase, analysts transform these

rules into more structured and detailed statements, e.g. *condition-action* statements. Single rule statements can yield more condition-action rules. The last phase is to design and transform rules into highly structured executable rules. Any statement that enforces the relation between data is considered a business rule. For the recognition of rules from the sentences of natural language a disciplines like data mining or text mining might be used.

The business rules approach manages the flow of business processes by using constraints or decision blocks. Business rules classify, compute, compare and control data to direct the flow.

The business rules can be:

- Restrictions – X must have Y
- Guidelines – X should have Y
- Computations – $X = f(Y)$
- Inferences – if X infer Y
- Timings – do X at time T
- Triggers – when X occurs do Y

or the combination of statements above [9].

Business rules' patterns can simulate the following types of events' behavior:

1. *logical operations* – conjunction (AND), disjunction (OR), negation (NEG);
2. *threshold patterns* – triggers when a threshold value has been processed;
3. *subset selection patterns* – selection of significant rules in a set;
4. *modal patterns* – check if assertion is true;
5. *time or spatial restrictions*, e.g. according to the spatial restriction, a payment fraud by credit card can be detected.

Note also that each business rule may be expressed in one or more formal rule statements, although each formal rule statement must be an expression of just one (atomic) business rule. A formal rule statement is an expression of a business rule in a specific formal grammar.

The business rule approach manages the flow of business processes by using constraints and/or decision blocks. Business rule patterns can simulate several types of events' behavior, such as logical operations, threshold patterns, subset selection patterns, modal patterns – check whether assertion is true, time or spatial restrictions – according to the spatial restriction, possible fraud can be detected. The classification of behavioral business rule types is presented in Figure 4.3. Behavioral rules can be further decomposed to support different patterns of implementation, depending on the granularity of the process implementation. Behavioral business rules express constraints or guidelines. Colors in the figure indicate different categories of business rules. The basic principles about this topisc are also summed up in [61].

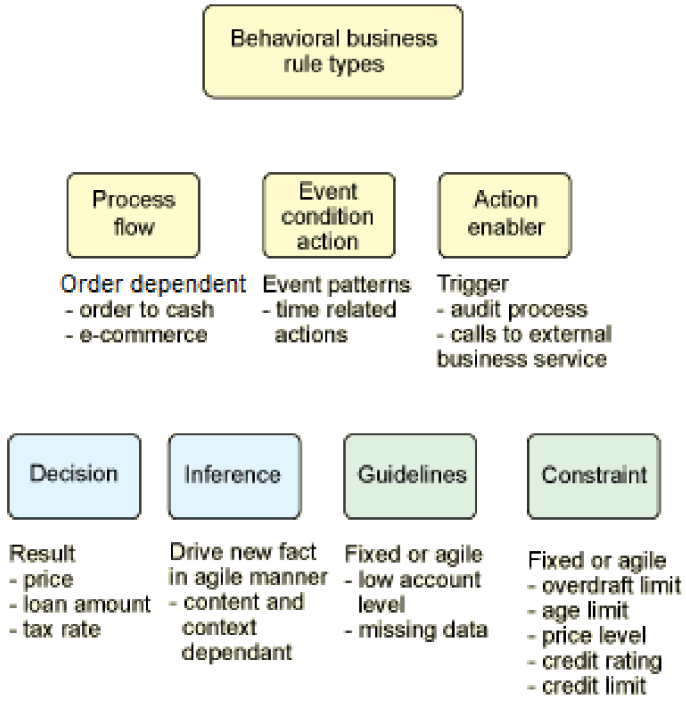


Figure 4.3: Business rules types – adapted from [5].

4.4.1 Use of Business Rules

Typical use of these rules is in processes that contain easily automated actions that do not depend on other human interaction or opinion. Rules can be used, for example, in algorithmic trading, e.g. for the placement of the order limit.

The following are two examples of business rules in practice:

Example. Execution of stock trading – limit order

1. If the price is less than $20 (*limit minimum price*), BUY stock from COMPANY.
2. If the price is at $35 (*limit maximum price*) or more, SELL stock from COMPANY.

These two rules are displayed in Fig. 4.4.

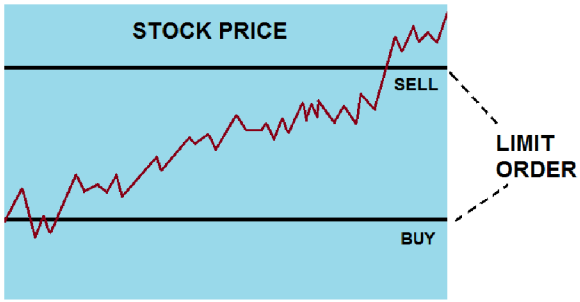


Figure 4.4: Example of Limit Order. Source: author.

4.5 Semantics of Business Vocabulary and Business Rules

Semantics of Business Vocabulary and Business Rules (SVBR) is a standard published by Object Management Group (OMG). In fact, SBVR is the OMG implementation of the business rules approach. This specification defines the vocabulary and rules for documenting the semantics of business vocabularies and business rules for the exchange of business vocabularies and business rules among organizations and between software tools. It is primarily conceptualized for business people – for the description of rules is mostly used natural language. SBVR is intended to formalize complex compliance rules, such as operational rules for an enterprise, security policy, standard compliance, or regulatory compliance rules. Such formal vocabularies and rules can be interpreted and used by computer systems.

SBVR is an integral part of the OMG's model-driven architecture (MDA). The SBVR standard defines the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules. SBVR can be further used to formalize complex compliance rules related to the software. SBVR provides a way to capture specifications in natural language and represent them in formal logic so they can be machine-processed.

Rule statements are expressed using either alethic modality or deontic modality and require elements of modal logic as formalization.

4.5.1 Alethic Modal Operators

In alethic logic, a proposition that is possible but not necessary is termed *contingent*. If people in a business were to treat it as a necessity, they would miscategorize things in the real world. This typically leads to refusal of activity (that should be permitted) because unnecessary preconditions are not met. SBVR Structural Business Rules use two alethic modal operators.

1. it is necessary that ?
2. it is possible that ?

4.5.2 Deontic Modal Operators

In deontic logic, a proposition that is permissible but not obligatory is termed "optional". If people in a business were to treat it as an obligation, they would demand compliance that is not required by the business. SBVR Operative Business Rules use two deontic modal operators:

1. it is obligatory that ?
2. it is permitted that ?

SBVR is primarily intended to be used for modeling in natural language. Based on linguistics and formal logic, SBVR provides a way to represent statements in controlled natural languages as logic structures called semantic formulations. SBVR is intended for expressing business vocabulary and business rules, and for specifying business requirements for information systems in natural language. SBVR models are declarative, not imperative or procedural. SBVR has the greatest expressivity of any OMG modeling language. The logics supported by SBVR are typed first order predicate logic with equality, restricted higher order logic (Henkin semantics), restricted deontic and alethic modal logic, set theory

with bag comprehension, and mathematics. SBVR also includes projections, to support definitions and answers to queries, and questions, for formulating queries. Interpretation of SBVR semantic formulations is based on model theory. SBVR has a MOF (Meta-Object Facility) model, so models can be structurally linked at the level of individual facts with other MDA models based on MOF.

SBVR specification defines a metamodel and allows to instance it, in order to create different vocabularies and to define the related business rules; it is also possible to complete these models with data suitable to describe a specific organization.

4.5.3 Adaptive Business Rules

In CEP, the processing takes place according to user-defined rules, which specify the relations between the observed events and the phenomena to be detected. We focus on event processing from the decision point of view. After the data have been processed by a CEP engine, we can distinguish recurring behavior in the data. These phenomena can be described by patterns. The designed model for decision making during the processing of data takes into account these patterns and so the system may react to this behavior and may apply the most suitable rule to data with which the process will continue. For example, if the trader has to decide when to buy or sell, we may apply the following rule:

- If `price_of_security` reaches `threshold_value` → open buy or sell position.

The next step in processing will be chosen according to information obtained from historical data and information given by the user. The user may update the behavior according to preconditions, which will be described in the system by business rules. The designed model of the system will allow the user to specify his or her own input rules [59].

Real-time processing is becoming an essential requirement and the use of adaptive set of rules may save time during processing. Subsequent steps in decision making may also be correlated by the model learning the meaningful sample of data. However, this is not the main aim of this thesis.

4.5.4 Business Rules and Events

A necessary step in understanding business rules and the business rules approach is to understand how rules are related to events. In the business rule approach, rules are always perceived and expressed *declaratively*, independent of processes and procedures. Generally, certain rules apply when certain events occur. But what exactly is the connection between rules and events? There are two perspectives how to answer this question [48].

The business perspective considers an event as something that requires a response from business. For example, a customer places an order. This is an event that requires a response. Response to this event can be given for example, within business process models, workflow models, procedures, and so on.

The perspective of information technology an event, or a change of state, is something that happens and needs to be recorded because the knowledge about the event may be critical to other business activities, either those occurring during the same time frame or those that might happen later.

In the business rule approach, recording of an event is always based on predefined terms and facts. An information system can support the structured business vocabulary in several ways (e.g. as a database design, a class diagram, and so on). The data that must be updated (created, modified, or deleted) to record the event are kept in the information system.

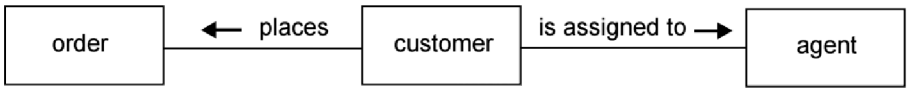


Figure 4.5: Terms and Facts for the Business Rule – adapted from [48].

A Fact Model, which is captured in Figure 4.5, is a static model which structures business knowledge about core business concepts and business operations. It is sometimes called a business entity model. The fact model expresses the core business concepts (called terms), and the logical connections between them (called facts). The facts are typically verbs which describe how one term relates to another. The business knowledge represented in a fact model should be at the most atomic level of business knowledge, meaning it should not be able to be further deconstructed and it cannot be derived from other knowledge. By using the standard vocabulary defined by the fact model, these basic building blocks can be used to develop and communicate more advanced forms of business knowledge, such as business rules, in a clear and unambiguous way.

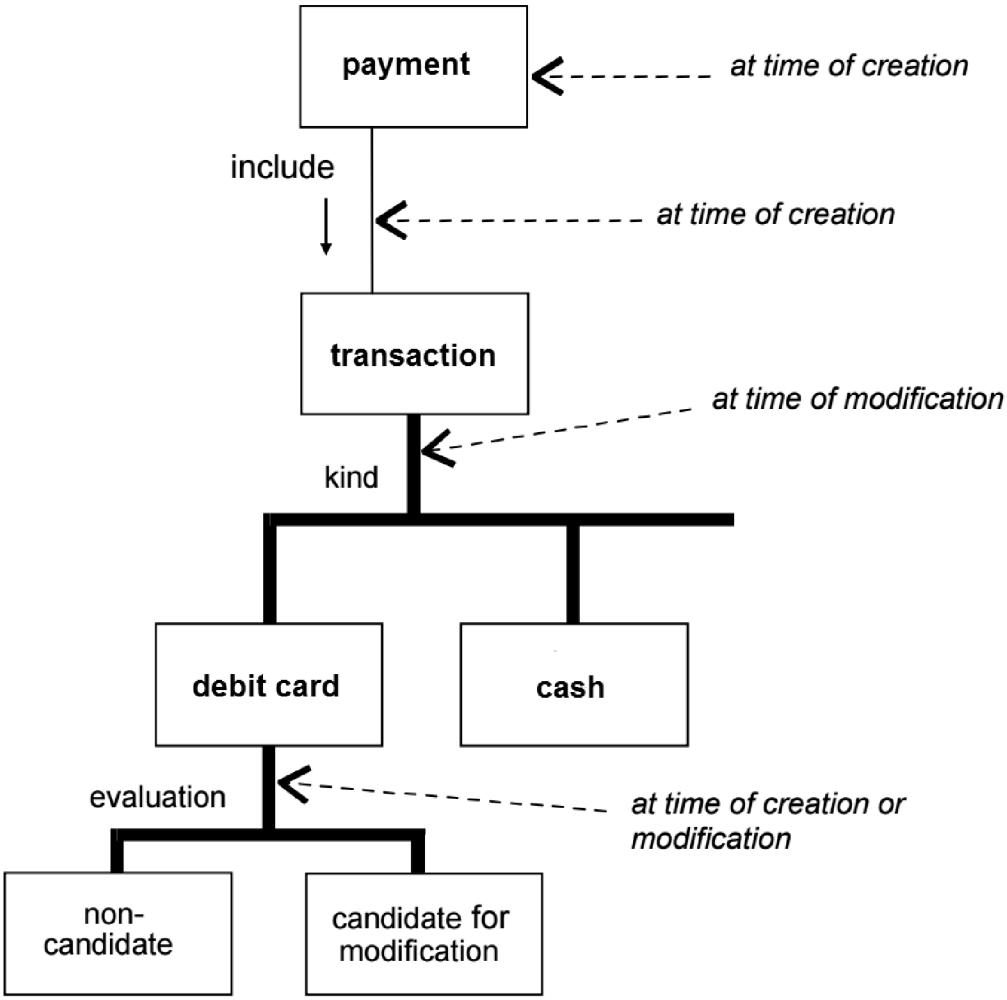


Figure 4.6: Multiple Events for Complex Rule – inspired by example from [48].

In Figure 4.6 is an example of the business rule of the payment. This rule produces

following events:

1. Update event 1: When an instance of payment is created.
2. Update event 2: When an instance of transaction is added.
3. Update event 3: When an instance of debit card payment transaction is included in a changes kind.
4. Update event 4: When an instance of debit card payment transaction already included.

Processes connect to rules via events. Processes produce events, which can fire one or more rules. The rules may determine whether the event is undertaken correctly or will produce a desired outcome. The rules are externalized from the processes and established as a separate resource. This permits direct management of the rules. The emphasis on rules and their separation from events and processes enables several opportunities:

- Simple Consistency – The two or more events are likely to be embedded in at least two and possibly more different processes. These events are represented by a single rule. That same rule should fire when any of the events occur in any of the processes. By this means, the business rule approach ensures complete consistency and/or decision logic applied across all the processes.
- Adaptability – Separating the rule from the events and processes allows the rule to be specified in one single source. The advantage for the implementation of one event is better for the code maintenance. The changes in the implementation can be done quickly in one place.
- Reengineering – Business processes and procedures are generally organized as responses to business events. For reengineering of business processes, the clarity of business rules enables better balance between action and guidance.

4.5.5 Business Rules and Decisions

This subsection describes how the business rules relate to decisions. Decisions are made through the decision services. They need to contain all the logic and algorithms necessary to make the decisions correctly. Generally, these services are stateless and they send responses to business questions to requesting services. Decision services typically have no side effects so they can be called whenever they are needed without any change in the system. This means that database updates, event generation or other actions taken as a result of the decision are taken by the caller not by the responding decision service.

Figure 4.7 shows the decision service together with other services that create an enterprise system. The role of desicion services can be summed up into several statements:

- The support of business processes by making the business decisions that allow a process to continue.
- The support of event processing systems by adding business decisions to event correlation decisions (represented by Event Processing Agents).
- The permission to be externalized for reuse and agility as crucial and high-maintenance parts of legacy enterprise applications.

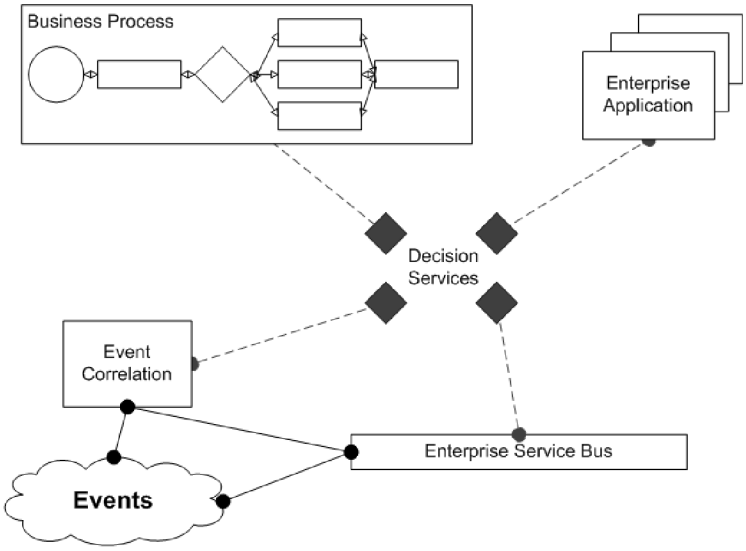


Figure 4.7: Integration of decision services – adapted from [51].

- By using Enterprise Service Bus approach the decisios servicec can be plugged into a variety of systems.

The design of decision service must be in accordance with several crutial characteristics. The behavior of decision services must be understandable to the business. As the business decisions may change even during the runtime, the decision service has to be both flexible and designed for the adaptive set of rules. They need an ability of integration of rules with historical set of data. The decision services must support multi-channel use so they can process more requests simultaneously. In order to respond sensibly when it cannot decide the services must manage all kinds of exceptions. It should be ensured that enough context is returned. Any decision service must be able to log exactly how it decided and that information must be accessible and readable also to non-technical users.

Each decision service, and thus each decision requires a number of sets of rules. These are coherent groupings of rules that can and most of the time should be used together. Some decisions require a single rule set and some require many. The relation between business rules and decisions is in Figure 4.8.

A single rule set decision might be represented by a decision table or a decision tree. The important thing is that the rules in a rule set execute as a set and get reused as a set.

A multi-rule set decision typically has a decision flow that lay out the steps involved in a decision, the branches and the loops and map the steps to specific rule sets. This is shown on the graphic as some decisions use more than one set of rules. This allows multiple decisions. Because rule sets are coherent sets of rules on a single topic, they typically have an obvious business owner – an individual or a group. This allows a clear separation of rule management (by rule set) from rule execution (by decision) – there is not one rule maintenance environment for each decision but one for each set of rules.

Rule changes are going to be made within a rule set and often multiple rules in a rule set will have to change in response to new changes. These rule set changes should be managed, tested, simulated and deployed. When they deploy they might alter the behavior of several decisions because they are reused in several rules sets.

Decisions are internally tied to events and processes in systems. The decisions are

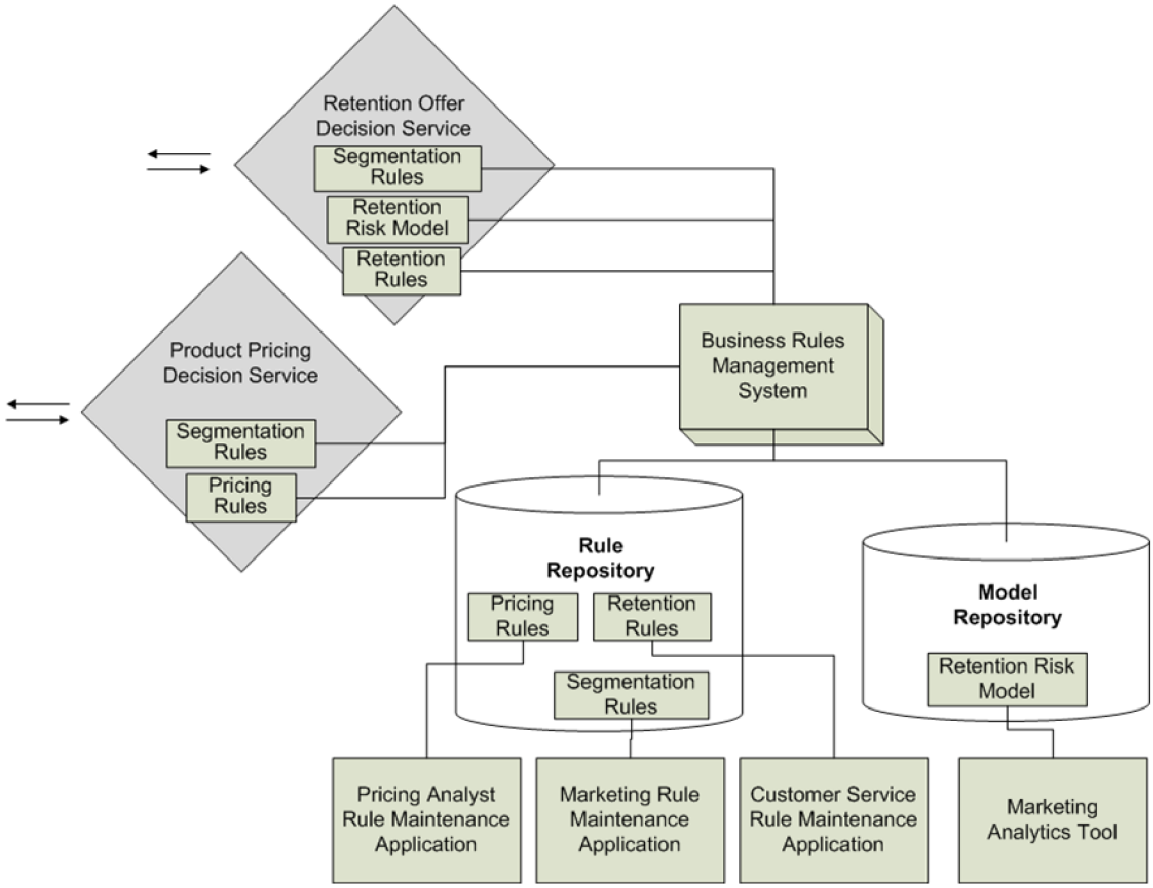


Figure 4.8: Relation between business rules and decisions – adapted from [51].

managed and maintained separately. Whether the decision service is a separate web service or an process function it can be accessed and updated separately. Overall, CEP technology can be seen as a means to serve as decision services.

4.5.6 Business Rules' Notation

Managing and modeling decisions are crucial for business. The DMN (Decision Model and Notation) standard emphasizes the importance of business decisions, and also offers a standard notation and expression for decision requirements and decision logic. A decision model based on the DMN standard is a graphical representation of a decision-making approach. Business analysts can model the rules that lead to a decision in a form of tables which can be executed directly by a decision engine. This approach minimizes the risk of misunderstandings between business analysts and developers, and it allows rapid changes in production [2] and [3].

4.6 Decision Tables

Decision tables are tools for:

- Capturing certain kinds of system requirements and knowledge.

- Documenting internal system design.

They are used to record complex business rules that a system must implement. In addition, they can serve as a guide for creating test cases. Decision tables represent complex business rules based on a set of conditions. The outcome may be multiple actions, not just one action. The notation of decision tables stipulates that the first column contains the labels of all evaluated facts and actions. Every other column contains one rule. Each cell in the table then indicates what value the fact has for a given row or what action should be taken.

Decision tables are clearly understandable both by the people who implement them into the decision support system or other system which helps to make decision and by the analysts who are the creators of the rules. The description of rules is declarative; it is a set of implications which is executed over the set of facts. This fixed order of conditions allows a complete overview of decision rules for a specific decision. It also allows grouping of related rules into tables, thereby providing an overview of a large number of decision rules. The decision table is one of the possible models for the description of business rules for the creation of DSS.

Trade decision table for the Buy order				
	Rule 1	Rule 2	Rule 3	Rule 4
Conditions				
Valid Symbol	No	Yes	Yes	Yes
Valid Quantity	DC	No	Yes	Yes
Sufficient Fund	DC	DC	No	Yes
Actions				
Buy?	No	No	No	Yes

Table 4.1: Placement of the Buy order – the example of a decision table for the Placement of the Buy order – based on [21].

The example in Table 4.1 shows the condition and the rules for the Buy order. As is seen, the only case when the Buy order is placed is when there are a *valid symbol* **AND** *valid quantity* **AND** *sufficient funds* available. Conditions that do not affect the outcome are marked "DC" for "Don't Care". So Rule 1 indicates that if the Symbol is not valid, ignore the other conditions and do not execute the Buy order.

Dictionaries of decision rules are another method of describing rules in DSS. Generally, the dictionary of rules is a set of facts, global values and functions. Another way to describe business rules is the enumeration of *IF condition THEN action*. The condition part of the rule is in the form *fact operator value* which is when the fact is tested whether it is equal, less or greater than the given value. All described methods are based on the form which is clear to both technical and nontechnical business people.

However, the real advantage for business is the ability to obtain consistency, completeness and correctness of the decision logic. Avoiding redundancy and overlapping rules is a key element in constructing and maintaining decision tables that offer value for business.

4.6.1 Implementation of Business Rules

The configuration of the decision support by business rules can be realized by an existing tool or framework. To mention the most used frameworks, we name JBoss Rules/Drools, Jess, and ILOG JRules. The idea of decision tables is directly implemented in the OpenRules framework [7]. OpenRules supports the OMG standard for business rules' notation – DMN. This framework uses the Excel sheet for definition and maintenance of the set of business rules. The set of rules can also be updated at the run time. OpenRules offers an enterprise level of business rules' repository implementation. Its advantage in comparison with the other tools listed above is that this framework is available as open source.

Chapter 5

Formalization of Business Rules: Design and Implementation

This chapter is divided into several parts. At first there is a brief review of state of the art of the different application areas using CEP and the problem it solves. This part is followed by the design of the method for the formalization of business rules. The method will be implemented as a module integrated in a form of decision support system into the complex event processing platform Esper. The second half of the chapter is devoted to the desription of implementation of DSS and complex event processing engine Esper. Esper was chosen because it is open-source platform among many other alternatives which natively implements CEP principles. This chapter also describes technologies on which the module is built on.

5.1 State of the Art

Currently, much information is provided in a form of data streams: sensors, software components and other sources are continuously producing fine-grained data that can be considered to be streams of data. Examples of application fields exploiting data streams are traffic management, smart buildings, health monitoring, financial trading, sensor monitoring, application logs processing, RFID tracking or smart grid measurements. Intelligent decision support systems in combination with advanced event processing analyze stream data in real-time to diagnose the actual state of a system allowing adequate reactions to critical situations [26].

For instance in traffic management, velocity measures must be related to specific knowledge about the road network (e.g. road topology and speed limits) and thus its data items are marked/enriched with semantic background knowledge.

Prototype of a system that provides end-to-end Quality-of-Service (QoS) management for mobile broadband telecommunications service operations using measurements collected from end-user devices is described in [15].

Other application of formalization is introduced in [34] where the approach is based on semantically rich event models using ontologies that allow the representation of structural properties of event types and constraints between them. Authors use a declarative approach to complex event processing that draws upon well established rule languages.

The authors in [12] and [54] deal with the approach of the distibuted CEP. The aspect of semantic actions in CEP is also discussed.

Another way of application is high-frequency predicting and the creation of a predicting model using neural network or genetic programming. There are many works investigating the high frequency data prediction by using methods from artificial intelligence area. In [37] two genetic system creating trading strategy are discussed. Other possible way in the field of artificial intelligence can be the use of genetic algorithms.

In [33] authors are dealing with the high-tech manufacturing industry problem where they solve the automated and timely detection of anomalies which can lead to failures of the manufacturing equipment.

Also, instead of building on top of existing engines, research usually revolves around creating new features in their own implementation. There are created solutions which fit a problem in a specific area, not new standards which can be used generally. There is still a lack of standardization of approaches in the area of complex event processing.

5.2 Formalization of Business Rules

This section is the core of this thesis. It introduces a new method for formalization of business rules which form knowledge base of decision-making component in CEP Esper. The particular rules will be described by using formal grammar with regulated rewriting. For the formalization we chose the matrix grammar as it allows to group rules into the multiple rules sets by grouping the labels of individual groups. This allows a separation of rule management from rule execution. Newly proposed method will optimize the current decision-making method in CEP.

The ruleset updates are going to be easier manageable within a rule set described by matrix grammar. The components of this grammar alows to group rules (rule labels) into multiple rules sets. When there are any changes deployed in classical approach they might alter the behavior of several decisions because they are reused in several rules sets. Within the use of matrix grammar we no longer have to copy the new rule into several sets but we put the new rule into one main set of rules, we label the new rule and we update the matrices with the labels. In case when we just update some existing rule and this rule is in several groups there is no need to update all the groups but we need to update just the main rule-set.

Business processes are key elements of all organizations, and their formalization enables the analysis of important functional and non-functional characteristics. A number of approaches for formalization of business rules exist, but, as far as it is known to the author of this thesis, none of them use formalism of a matrix grammar. In the designed approach, we take advantage of the main characteristics of matrix grammars which are the generative power, ease of use and the good maintenance of the set of rules. The user may update the set of rules as required, without the need of a third party to control the decision-making process. CEP decision making is stateful, so we use the information from the previous states.

To mention other approaches, in [39] authors present formalization of business rules based on ontology and UML modeling with the use of Object Constraint Language.

The logic-based approach of the use of language for event processing is introduced in [52]. Authors propose a homogeneous reaction rule language for complex event processing. It is a combinatorial approach of event and action processing, formalization of reaction rules in combination with other rule types such as derivation rules, integrity constraints, and transactional knowledge.

In [31], the idea of formalization of business rules with the use of grammatical systems is discussed.

5.2.1 Matrix Grammar

Formal grammars can be used for the description of behavioral patterns and the set of business rules extracted by CEP and for the support of prediction of data in CEP platforms. Briefly, a formal grammar is a set of rules for rewriting strings, along with a "start symbol" from which rewriting starts. Matrix grammar belongs to the group of regulated rewriting grammars. For further reading on this topic, the authors recommend [49]. The definition of matrix grammar follows.

Definition 4 *Matrix grammar is a pair* $H = (G, M)$*, where* $G = (N, T, P, S)$ *is context-free grammar and* M *is finite language over* $P, (M \subset P^*)$ *– sentence of this language is called* matrix.

Formally, a matrix grammar is a pair $H = (G, M)$*, where*

1. $G = (N, T, P, S)$ *is a context-free grammar, where:*
   (a) N *is an alphabet of nonterminal symbols.*
   (b) T *is an alphabet of terminal symbols.*
   (c) P *is a finite set of rules,* $P \subseteq N \times (N \cup T)^*$.
   (d) S *is starting symbol,* $S \in N$.
2. M *is a finite language over* $P, (M \subset P^*)$ *– a sentence of this language is called a matrix.*

Further, for $u, v \in (N \cup T)^*, m = p_1 \dots p_n \in M$ *we define* $u \Rightarrow v[m]$ *in* H*, if there are strings* $x_0, \dots, x_n$ *such that* $u = x_0, v = x_n$*, and for all* $0 \leq i < n, x_i \Rightarrow x_{i+1}[p_{i+1}]$ *in* G.

The language generated by H*, denoted by* $L(H)$*, is defined as*

$$L(H) = w : w \in T^*, S \Rightarrow^* w.$$

Even though matrices contain only context-free rules, they generate a *context-sensitive* language. The example of matrices upon a main ruleset can be found in Figure 5.1, where particular matrices are highlighted.

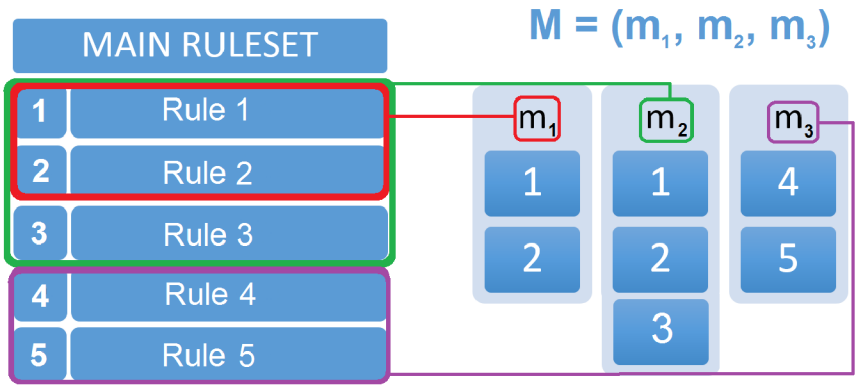


Figure 5.1: Sample ruleset of decision support component and matrices mapping. Source: author.

5.2.2 Formalization of Business Rules by Using Matrix Grammar

Input: Business rules in various forms. Business rules can be in the form of decision tables, enumeration of condition-action rules, or sentences in natural languages. The form of business rules is discussed above. In this example, we refer to the general form of the decision table above – Table 4.1

Output: DSS described by the business rules in the form of matrix grammar $H = (G, M), G$ is quadruple (N, T, P, S)

Method:

initialization;

$N := \{Action_1, Action_2, \ldots, Action_n\}$;

$T := \{condition_1, condition_2, \ldots, condition_m, action_1, action_2, \ldots, action_n\}$;

$P := N \times (N \cup T)^*$

foreach *$Rule_p$, where $Rule_p \in \{Rule_1, Rule_2, \ldots, Rule_p\}$ from the decision table consider all suffice conditions, the set $\{Condition_1, Condition_2, \ldots, Condition_m\}$* **do**

1. add rule $p, p \in P : S \rightarrow < Rule_1, Condition_1 >< Rule_1, Condition_2 > \cdots < Rule_1, Condition_m >$;
2. add rule $p, p \in P : S \rightarrow < Rule_2, Condition_1 >< Rule_2, Condition_2 > \cdots < Rule_2, Condition_m >$;
3. $\ldots$;
4. add rule $p, p \in P : S \rightarrow < Rule_p, Condition_1 >< Rule_p, Condition_2 > \cdots < Rule_p, Condition_m >$;
5. add $< Rule_p, Condition_m >$ to N; m, p are positive integers.

end

foreach $< Rule_p, Condition_1 >$ **do**

add rule:

- $< Rule_p, Condition_1 > \rightarrow Action_n condition_1$
- $Action_n \rightarrow action_1 action_2 \ldots action_n$, where $action_n$ are all actions taken after fulfilling of all sufficient conditions for the $Rule_p$.

end

foreach $< Rule_p, Condition_m >$ **do**

add rule:

- $< Rule_p, Condition_m > \rightarrow condition_m$

end

$S := S$; (~waiting state);

$M := \{m_1, m_2, \ldots, m_p\}$, where $m_p = [< Rule_p, Condition_1 > Action_n condition_1, < Rule_p, Condition_m > \rightarrow condition_m$ for all $m > 1]$;

Algorithm 1: Formalization of Business Rules By Using Matrix Grammar

Component M is usually created by a business analyst by determining parallel actions. In this case, the matrices are determined by the grouping of all conditions into a matrix

and all actions into one matrix. All rules in each matrix have to be executed in one computational step. The basic idea of the formalization and the introduction of this method was published in [61], [62] and [63].

5.2.3 Quote Data

As sample data, we took historical financial high-frequency data from Forex. Data are available at [10] where samples of high-frequency data sets from different exchanges can be downloaded. For testing purposes, these data are adequate. Quote data from markets are provided in CSV files which contain the following formats:

```
07/08/2013,16:00:00.113,1.28711,1.2872,TDF,,LAX
07/08/2013,16:00:00.269,1.28704,1.28729,FXN,,NYC
07/08/2013,16:00:00.269,1.28704,1.28729,FXN,,
07/08/2013,16:00:00.348,1.28706,1.28726,SAXO,EUR,CPH
07/08/2013,16:00:00.414,1.28708,1.28725,GACI,NAM,NYC
07/08/2013,16:00:01.031,1.28707,1.28726,GACI,,
07/08/2013,16:00:01.164,1.287,1.2874,CCIB,ASI,LBU
07/08/2013,16:00:01.195,1.2871,1.28723,GOSP,,SGP
07/08/2013,16:00:01.250,1.28712,1.2872,TDF,NAM,LAX
07/08/2013,16:00:01.250,1.28712,1.2872,TDF,,
07/08/2013,16:00:01.250,1.28712,1.2872,TDF,,
07/08/2013,16:00:01.250,1.28712,1.2872,TDF,,
07/08/2013,16:00:01.278,1.28708,1.28726,GACI,,NYC
```

Each row expresses one record – event – and contains variables which meaning is summed in the following Table 5.1.

Table 5.1: Quote Data Format – adapted from [10].

Quote Data Format		
Field Name	Data Type	Description
Quote Date	MM/DD/YYYY	Date of Quote
Quote Time	HH:MM:SS.000	Time of Quote in milliseconds
Bid Price	Number	Bid price
Ask Price	Number	Ask price
Contributor Code	String	Feed Source
Region Code	String	Region where feed source is located
City Code	String	City where feed source is located

If we take for example record:

```
07/08/2013,16:00:00.348,1.28706,1.28726,SAXO,EUR,CPH
```

then we read it as folows: On 07/08/2013 at 16:00:00.348 was the bid price of the quote 1.28706 and the ask price of the qoute 1.28726. Quote appeared at Saxo Bank (SAXO) in Europe (EUR) in the town of Copenhagen (CPH).

The input stream of data is in this format and it is daved in CSV file.

5.3 Esper

The following text is mainly based on information from the Esper Reference Documentation [4] and the web pages of this tool [46].

The CEP module integrated with proposed method of decision making is build by using the engine Esper, a powerful open-source engine for CEP that provides powerful Event Pattern Language (EPL) for complex events detection. ESPER engine works a bit like a database turned upside-down. Instead of storing the data and running queries against stored data, the ESPER engine allows applications to store queries and run the data through them. The CEP engine gives responses whenever the conditions occur that match queries. The execution model is thus continuous rather than only when a query is submitted. All of Esper computing is in-memory computing. The latency of Esper is usually below $10\mu s$ with more than 99% predictability.

5.4 Design and Implementation details

Esper is an open source engine and it is suitable for integration into many environments and software products. Esper combines both the CEP approach and event stream processing (ESP). ESP queries involve simple select queries and window aggregations on a single stream of data. CEP is a super set of ESP. Differences between ESP and CEP are discussed in [41]. In CEP, we find patterns, derive new events based on a combination of input events, possibly from multiple streams of data.

Esper is available in Java or in C# .NET as NEsper. This platform enables rapid development of applications that analyze high-frequency data, combining historical and real-time data. Esper filters and analyzes events in various ways and responds to conditions of interest. Esper provides a rich declarative language for dealing with high-frequency time-based event data for pattern definition called Event Pattern Language (EPL). EPL is SQL-based and offers all SQL operators extended with temporal operators. Spatiotemporal patterns are defined in the ESPER knowledge base pattern and they are used by the pattern matching process. The goal of CEP is to identify meaningful events (opportunities or threats) and respond to them as quickly as possible [46]. The basic principles about this topisc are also summed up in [61].

Esper uses indexes, a data structure that improves the speed of data retrieval operations. For sorted access it may prefer a binary tree index while a hash-based index is great for key lookups. For efficient matching of incoming events to statements the engine uses inverted indexes. Multi-version concurrency control is a concept used for variables and also for filters to allow concurrency and reduce locking.

The match-recognize pattern matching functionality is built using nondeterministic finite automata (NFA). Query planning based on the analysis of expressions used in the where-clause is another technique used by the engine. The execution strategy may choose nested-loops versus merge joins. The Esper grammar is built using ANTLR and based on Extended Backus-Naur Form (EBNF). Allan's interval algebra is the foundation for many of the date-time methods. Enumeration methods employ lambda expressions – closures [46].

Memory use depends on the statements and the memory used by events. Esper keeps the minimal information needed to satisfy a statement in memory, and also can share data windows between statements. Esper offers built-in data windows as part of the event processing language that instruct the engine how many or how long events must be considered.

For example, a time window with an interval length of 10 seconds instructs the engine to retain the last 10 seconds of events as a moving data window. For example, if the statement employs no data window, Esper keeps no events in memory.

Esper also re-uses (shares) data windows between statements, if possible. For views that derive values from an event stream, no events are kept in memory. For aggregations, only the aggregation values are kept in memory. For patterns, the events that participate in the pattern are kept in memory only if tagged. For output rate limiting, events are buffered unless you use the „snapshot“ keyword - There is a section in the output limiting doc explaining what is buffered and when for output rate limiting. For joins, if no data window is specified, the keep-all data window applies.

5.4.1 Applications using Esper

Examples of applications using Esper are:

- Business process management and automation (process monitoring, BAM, reporting exceptions, operational intelligence)
- Financial instruments (algorithmic trading, fraud detection, risk management)
- Network and application monitoring (intrusion detection, SLA monitoring).
- Sensor network applications (RFID reading, scheduling and control of fabrication lines, air traffic)

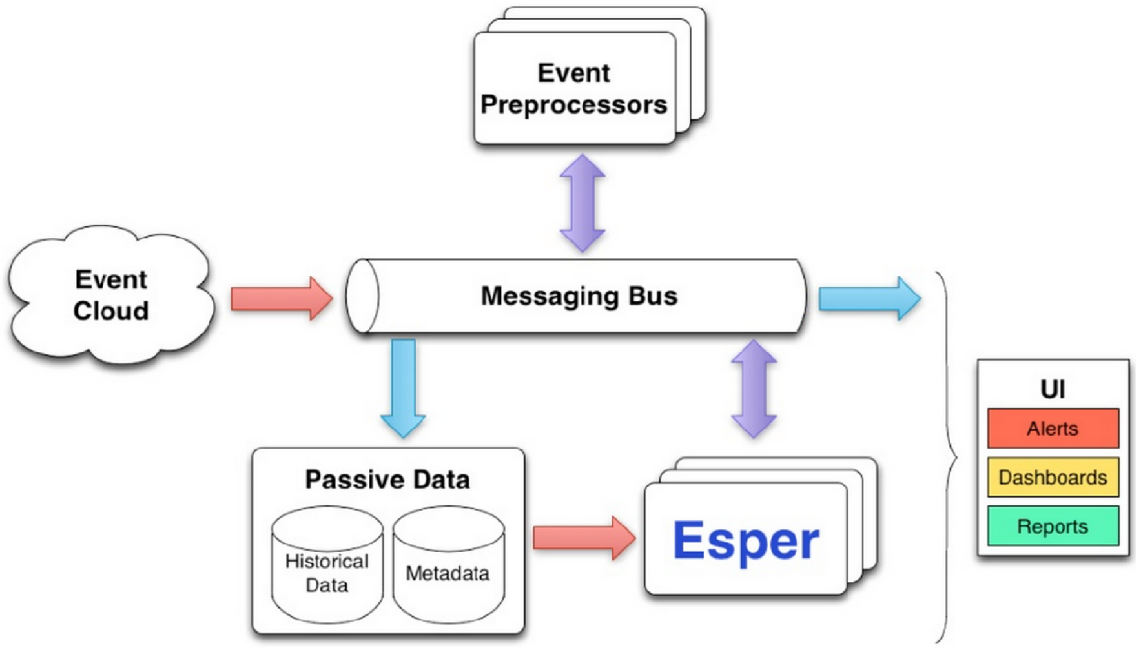


Figure 5.2: Schema of Esper platform – adapted from [1].

Figure 5.2 shows a schema of an Esper platform and Figure 5.3 shows a core of the Esper engine. The Esper engine is based on the use of state machine technology which is intuitive. We find this feature interesting and quite simple for integration – in comparison with other tools — with the model of the set of business rules controlled by matrix grammar. Esper

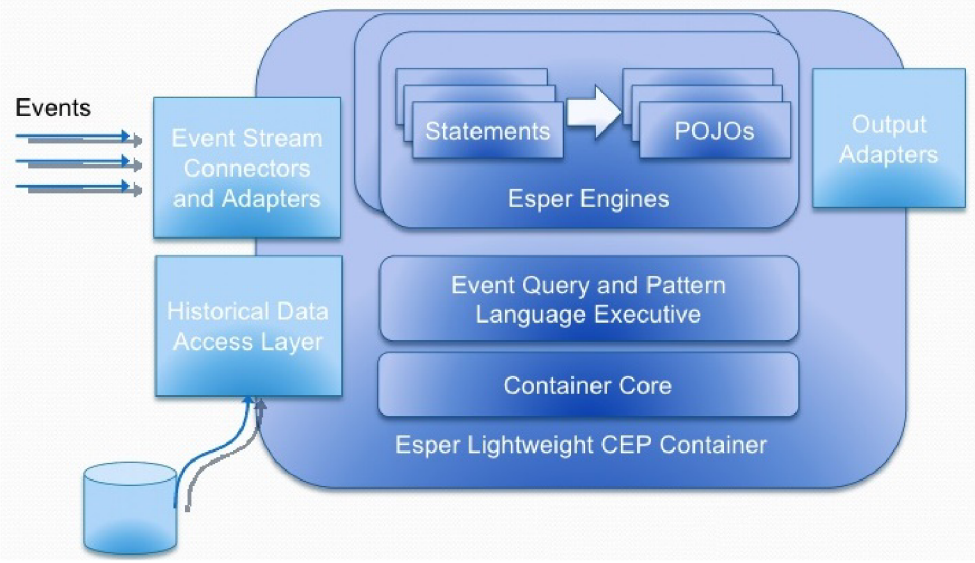


Figure 5.3: Core of Esper CEP platform – adapted from [41].

includes a historical data access layer to connect to most of the common databases and it is also possible to combine historical data and real-time data in one single query. Esper can easily be integrated with most available servers (Weblogic, Websphere, JBoss, Tomcat, etc.), service buses, grid platforms, and Microsoft-based .Net technologies for NEsper. This platform supports different kinds of input event formats, from Java / .Net objects and maps to XML documents. The Esper engine includes failover and recovery capabilities, ensuring that the engine is usable non-stop (high availability). Another advantage is the custom adding of event storage options. As performance tests show Esper scales vertically nearly linearly (by adding more CPU power). In a VWAP (Volume Weighted Average) benchmark Esper exceeded 500 000 events per second on a dual CPU server class hardware, with only 5 microsecond average latency. Horizontal scaling is best handled by logical partitioning of statements and data streams to separate Esper instances [46].

Esper offers work with a time-based batching window, for example, combining events for specific time window size (1min, 30seconds, etc.). This feature is very important for the decision-making process, e.g. for threat detection. For example, if events can be batched for the previous 1 minute and a fault can be found within this time window, it can be predicted immediately. For a real-life problem, the size of the time window needs to be set very precisely. The Esper CEP maintains a batch buffer to keep all the events coming into the Esper [1]. Batch buffers also serve as a means of coping with network distribution issues: a business platform that generates a lot of events that need to be consumed by many clients might choose to group these events by a time unit to keep the network stress level low, instead of distributing these events one by one.

Esper's advantage is that it is open-source software. In comparison with other CEP, it does not have as many tools as i.e. StreamBase provides, but its strength is in the core engine that is embeddable into third-party solutions. Most of the upcoming information were adapted from Esper web pages and for more details about Esper tool author recommends [46] where the actual tutorial is also available.

5.5 Event Representations

An event in Esper is an immutable record of a past occurrence of an action or a change of state. Every event contains properties which carry information about a given event. Even the property itself can represent an event, and such a property is called a fragment. The Esper engine needs to know what the events look like (the name of the event, its properties, data types, etc.) before it can process them. The user can predefine events at the start-up via a configuration file or during runtime via API or EPL syntax. In Esper, events are represented by Java objects. Thus as the implementation language was chosen Java. In Esper, an event can be represented by any of the following Java objects as summed up in Table 5.2

Esper: Event representation	
java.lang.Object	Any Java POJO (plain-old java object) with getter methods following JavaBean conventions; Legacy Java classes not following JavaBean conventions can also serve as events.
java.util.Map	Map events are implementations of the java.util.Map interface where each map entry is a propery value.
Object[]	Object-array events are arrays of objects (type Object[]) where each array element is a property value.
org.w3c.dom.Node	XML Document object model.
org.apache.axiom.om.OMDocument	XML – Streaming API for XML (StAX).
Application classes	Plug-in event representation via the extension API.

Table 5.2: Event Underlying Java Objects [4].

Event properties capture the state information for an event. Event properties can be simple as well as indexed, mapped and nested event properties. The table below (Table 5.3) outlines the different types of properties and their syntax in an event expression.

Any expression can be used as a mapped property key or indexed property index by putting the expression within parenthesis after the mapped or index property name

5.6 Esper Features

5.6.1 Data Windows

Data windows are for managing fine-grained event expiry. They instruct the engine how long to retain relevant events or under what conditions events can be discarded. Data windows operate on the level of individual queries, streams and subqueries. For example, using a slide time window to keep arriving events for N seconds. The engine let events go (expires) that are older than N seconds. Having a good variety of configurable and combinable

Type	Description	Syntax
Simple	a property that has a single value that may be retrieved.	name
Indexed	An indexed property stores an ordered collection of objects (all of the same type) that can be individually accessed by an integervalued, non-negative index (or subscript).	name[index]
Mapped	a mapped property stores a keyed collection of objects (all of the same type).	name(key)
Nested	a nested property is a property that lives within another property of an event.	name.nestedname Syntax

Table 5.3: Types of Event Properties [4].

data windows available allows to address more analysis requirements and address common requirements concisely. Sliding window has several parameters, among them are i.e. time, length, sorted, ranked or ordering of data.

5.6.2 Named Windows

Named windows are globally visible data windows that allow sharing sets of events between queries efficiently, removing the need to keep the same events in multiple places. They define custom criteria for entering events and for expiring events. They also allow defining event expiry once and apply it across multiple queries.

5.6.3 Tables

Tables are global data structures that can hold aggregation state alongside data and events, and allow update-in-place. Table columns can be of type aggregation and table rows can hold the aggregation state itself. Tables allow statements to colocate aggregation state, update-in-place data and event data conveniently.

Esper supports on demand queries against tables including joins. Esper supports explicit indexes, update-insert-delete and select-and-delete in a single atomic operation. Allows multiple statements to aggregate into the same state (coaggregation).

5.6.4 Event Pattern Language

EPL is a powerful and complex language. The reader can find more information about this language in Esper documentation [4].

As previously mentioned, EPL is a language for expression-based pattern matching and querying of data streams. EPL language syntax is similar to SQL, with clauses like SELECT, FROM, WHERE, GROUP BY, HAVING and ORDER BY. However in EPL, event streams replace data tables and the basic data unit is an event, not a data row. Because events are also data, the concept of joining, filtering and aggregation is leveraged in EPL in the same way as in SQL. Not every clause has the same meaning as in SQL. For example, an INSERT INTO clause is used for forwarding events to another stream or for joining multiple streams into one. An UPDATE clause is used for updating values in event properties. Expressions in EPL are written in the form of EPL statements. EPL statements are divided into two main types: EPL queries and EPL patterns. Both types are the implementation of event-processing mechanisms, mentioned in the previous chapter. EPL queries represent event stream queries and EPL patterns represent event patterns.

5.6.5 EPL Queries

EPL queries provide filtering, joining, grouping and aggregation of features. They also support the creation of window views and application of function upon event streams. For instance, EPL queries can join events from multiple event streams into a single stream, select specific properties of an event stream, compute an aggregation function (count, sum, max) over events which enter the event stream at the last minute, and much more. EPL queries are stored in the engine and they publish results for listeners when events match the criteria specified in the given query. The listener has to be attached to the EPL query via API before the EPL query is started. The most used and important clauses in EPL queries are SELECT, FROM, WHERE. They have meanings similar to those in SQL. The SELECT clause specifies which event properties or events are retrieved by the query. The FROM clause specifies the names of the streams from which the query reads. The WHERE clause specifies the search conditions according to which events are filtered. An example of simple SELECT-FROM-WHERE can be:

```
select avg(price) from StockTick.win:time(30 sec) where symbol='IBM'
```

5.6.6 Administration

The setup of the Esper engine consists of a few lines, most of them are optional. The only necessary code for creation of an instance of the Esper engine class is provided by:

```
EPServiceProvider epService =
EPServiceProviderManager.getDefaultProvider(config);
```

where `config` is optional and an instance of the `Configuration` interface which can be used to register custom events in the form of JavaBeans with the engine. By using the command `getProvider(uri_name)` instead, a unique instance of the engine will be returned. The same instance can be retrieved with subsequent calls using the same `uri_name`. Because of this it is not necessary to store/cache an engine inside of the calling program.

To free up some resources when using multiple engines an existing engine can be destroyed with the command

```
epService.destroy();
```

An engine instance that has already been used before and is now retrieved again with the `getProvider(uri_name)` command can be initialized again with the `epService.initialize();` command. The commands of Esper's EQL can be issued by requiring an administration

instance through the EPAdministrator interface from the engine instance like the following example shows:

```
EPAdministrator admin = epService.getEPAdministrator();
```

Then an instance of EPStatement can be requested from the `EPAdministrator` instance through two different methods by providing the statement that should be issued to the engine in form of a string which will then be parsed and interpreted:

```
EPStatement myStatement = admin.createEPL("EQL command");
EPStatement myTrigger = admin.createPattern("EQL pattern");
```

Optionally a name in form of a string can also be send to the engine to identify it at a later point by it. The returned EPStatement instance is the representation of the issued command inside of the superordinate program. Everything that happens inside of the Esper engine is connected to this instance as will be shown in more detail in the following paragraphs.

While all of Esper's clauses can be used with the `createEPL()` the engine instance also has the ability to create statements via its EPStatementObjectModel. This model is an objectoriented representation of an EPL or pattern statement and can be constructed through several methods provided by the class. It can then be used in a create method call instead of the textual representation of the statement that was used in the former example.

The EPStatementObjectModel can be transformed to a text statement and vice versa with methods supplied by `EPAdministrator`. After the creation of an `EPStatement` instance through a create method the statement's clause will be immediately inserted into the engine and is active from that moment. To manually stop its execution and restart it afterwards Esper provides the following two methods:

```
myStatement.stop();
myStatement.start();
```

5.7 Decision Support System Implementation

Following text describes an implementation of the Decision Support System (DSS) prototype application to prove the feasibility of the designed solution. A decision support system could contribute to this work by identifying important information. However, it is essential that when used in real system the information always should be approved and updated by an analyst.

A module of the DSS for the support of decisions was implemented in order to test the proposed method of formalization. The use of rules in this module is controlled by the matrices of business rules. The present system for real-time situation assessment and decision support comprises three main modules which can be found in Figure 5.4:

1. Event Stream Filtering of input data within the knowledge of business rules.
2. Event Stream Aggregation of input data within the knowledge of business rules.
3. The central knowledge base for decision support and knowledge management. The knowledge-based module runns rules described by the matrix grammar.

First two modules are the real-time module running an Esper statements. The approach takes advantage of complex-event processing engine Esper for analyzing the data streams. The main task of Esper is to filter and aggregate relevant information within the input flow of events.

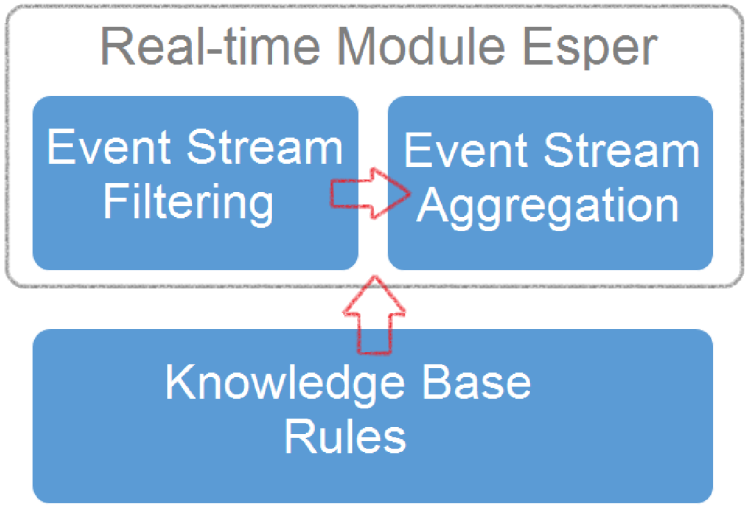


Figure 5.4: Schema of implemented DSS. Source:author.

5.7.1 Knowledge Base

Raw data is received in real-time from the feed file with FOREX historical data. Data are aggregated and refined and the new information is added to the knowledge base of the DSS. The rules and the patterns of rules are then exctracted from the input stream of data. These rules are used for the pattern matching and creates the knowledge base of proposed decision support system.

5.8 Econometric requirements

Econometrics is defined as economic theory in its relation to statistics and mathematics and its object as the unification of the theoretical-quantitative and the empirical-quantitative approach to economic problems. The central problem has been how precisely to combine economic theory, mathematics and statistics. Econometrics uses a combination of economic theory, math and statistical inferences to quantify and analyze economic theories by leveraging tools, such as frequency distributions, probability and probability distributions, statistical inference, simple and multiple regression analysis, simultaneous equations models and time series methods.

According to the [20] financial econometrics can be useful for testing theories in finance, determining asset prices or returns, testing hypotheses concerning the relationships between variables, examining the effect on financial markets of changes in economic conditions, forecasting future values of financial variables and for financial decision making. A real-life application of financial econometrics can be e.g. measuring and forecasting the volatility of bond returns, testing technical trading rules to determine which makes the most money, or testing whether spot or futures markets react more rapidly to news. The hypothesis can be tested and proven using econometric tools, such as frequency distributions or multiple regression analysis.

Econometrics was pioneered by Lawrence Klein, Ragnar Frisch and Simon Kuznets.

5.8.1 The Methodology of Econometrics

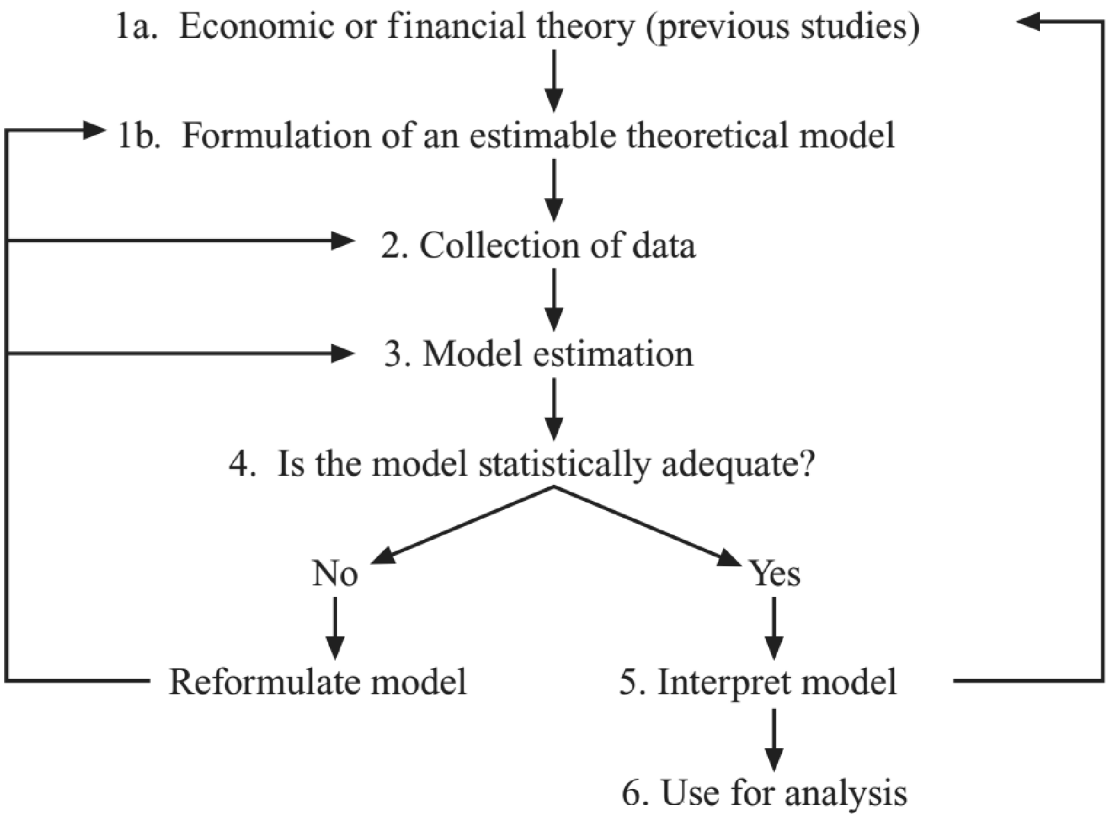


Figure 5.5: Schema of econometric model forming – adapted from [20].

Many different ways of describing the process of model building exist. Econometrics uses a fairly straightforward approach to economic analysis. In the following text, we describe the steps as described in [20]. A logical and valid approach would be to follow the steps illustrated in Figure 5.5.

- *Step 1a and 1b: general statement of the problem*
  This will usually involve the formulation of a theoretical model, or intuition from financial theory. The explanatory variables being analyzed are specified during this step; the relationship between the dependent and independent variables are also specified. The model is unlikely to be able to completely capture every relevant real-world phenomenon, but it should present a sufficiently good approximation that is useful for the purpose at hand.

- *Step 2: collection of data relevant to the model*
  The set of data required may be available electronically through a financial information provider, such as Reuters or from published government figures. Alternatively, the required data may be available only via a survey after distributing a set of questionnaires i.e. primary data. The next step is to define a specific hypothesis that explains the nature and shape of the set. This stage of econometrics relies on economic theory that will be tested for validity in the later stages.

- *Step 3: choice of estimation method relevant to the model proposed in step 1*
  For example, is a single equation or multiple equation technique to be used?

- *Step 4: statistical evaluation of the model*
  What assumptions were required to optimally estimate the parameters of the model? Were these assumptions satisfied by the data or the model? Also, does the model adequately describe the data? If the answer is 'yes', proceed to step 5; if not, go back to steps 1–3 and either reformulate the model, collect more data, or select a different estimation technique that has less stringent requirements.

- *Step 5: evaluation of the model from a theoretical perspective*
  Are the parameter estimates of the sizes and signs that the theory or intuition from step 1 suggested? If the answer is 'yes', proceed to step 6; if not, again return to stages 1–3. An effective model outlines a specific mathematical relationship between the explanatory variable and the dependent variable being tested. The most common relationship is linear, meaning that any change in the explanatory variable will have a positive correlated with the dependent variable. This is why the multiple linear regression model is the most used tool in econometrics, because it expresses relationships linearly.

- *Step 6: use of model*
  When we are finally satisfied with the model, it can then be used for testing the theory specified in step 1, or for formulating forecasts or suggested courses of action. This suggested course of action might be for an individual (e.g. "if inflation and GDP rise, buy stocks in sector X"), or as an input to government policy (e.g. "when equity markets fall, program trading causes excessive volatility and so should be banned"). This step validates the proposed model and its hypothesis. The test will help in the understanding whether or not the model resulted in good predictions or not. If it is observed that the results are as expected, then we may assume that the hypothesis is true. If the result is not as expected, new hypotheses or inferences are needed.

The process of building a robust empirical model is iterative, and it is certainly not an exact science. This means that the final preferred model could be different from the one originally proposed, and need not be unique. It is not guaranteed that two different researchers with the same data and the same initial theory will arrive at the same final specification.

The econometric requirements are proved by testing the scalability parameters. The whole econometric model is not additionaly created as the testing of crutial input parameters is sufficient. In spite of that all the steps are included thgoughout this thesis.

Chapter 6

Measurements and Experimental Results

Even though the validation may be done theoretically, it is important to take experiment measurements and results with real implementation of the theoretical model in order to validate more complicated properties of the system.

The sample dataset for a statistical test is selected just from out-of-sample data. This data is from a time period that has no overlap with the time period in which the model for prediction is developed. If the data have played any role in the development of the model of the system, any statistical test of its performance will be distorted. Data are restricted to objective methods that can be simulated on historical data.

Backtesting methods produce historical performance statistics which are evaluated in a statistically rigorous way. Profitable past performance is not taken at true value but it is rather evaluated as the possibility that backtest profits can occur by coincidence. The problem of this performance is especially pronounced when many methods are backtested and the best method is selected. This activity is called data mining. Though data mining is a promising approach for finding predictive patterns in data produced by largely random complex processes such as financial markets, its findings are upwardly biased. This is the data mining bias.

The biggest problem of all optimization techniques is that they optimize trading strategies regarding a given sample of market data. These data are often referred as training or in-sample. An optimization algorithm finds the optimal solution, but it doesn't evaluate how stable this solution will be if market conditions change.

Because of this, the successfulness of trading strategies is often tested on the out-of-sample (testing) data, which may potentially contain different market conditions. The optimized trading strategy can be considered as robust if its out-of-sample performance has the same characteristics as the performance based on the training data.

At first, financial markets are very dynamic places, where it is impossible to continually gain the same profits with unmodified trading methods over a long period of time. The optimization of trading strategies on the in-sample data is, in fact, the extrapolation of past into the future.

Modeling and simulation of automated trading strategies allows continuous repeating and comparing the performance of various strategies on the same data. This process gives an opportunity for introducing of new optimization methods which improve the strategies. The optimization of trading strategies is a process of making the strategies more profitable,

robust and stable considering the given market conditions.

6.1 Robustness

In financial economics, *robustness* is the ability of a financial trading system to remain effective on different markets and under different market conditions, or the ability of an economic model to remain valid under different assumptions, parameters and initial conditions.

In general, being robust means a system can handle variability and remain effective. A trading model is considered robust if it is consistently profitable when applied to various securities and in all market conditions including up-trends, down-trends, and range-bound markets. Very often, a trading model will function very well in a specific market condition or time period. However, when market conditions change or the model is applied to another time period or the future, the model fails.

Definition 5 *ANSI and IEEE define robustness as the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. Robustness can be defined as "the ability of a system to resist change without adapting its initial stable configuration".*

Since simulation tools allow traders to repeatedly and comparatively run various strategies on the same data, they are appropriate also for a wide spectrum of optimization tasks. The optimization of trading strategies is a process of making them more profitable, robust and stable considering given market conditions. Automated trading systems are based mainly on a computational evaluation of crisp inputs. Even though this evaluation is an objective process that eliminates psychological aspects of trading, it is not very robust since it doesn't reflect well the vagueness of the real trading environment. This is caused by a fact that conventional automated trading strategies are primarily based on an exact and symbolic representation of the reality [42].

The human factor in a semi-automated trading system improves its robustness and eliminates many errors caused by the fully automatic system. The robustness of the implemented system is partly ensured by the analyst who specifies the groups of business rules into matrices.

6.2 Out-of-Sample Testing

Out-of-sample testing is a way to guard against *curve-fitting*. Curve-fitting in general is the process of finding the (mathematical) description which best matches a given set of data. When it is not applied to trading strategies, it can be a very useful way of drawing conclusions from experimental data.

When applied to trading strategies, curve-fitting can produce over-optimized, over-optimistic results. In any set of price data, there is some "magic" combination of indicators and parameters that catches most every move and shows outstanding results. Unfortunately, that magic formula is a result of chance and is different for every data set. That means that future results probably won't come close to the numbers generated with the full benefit of hindsight. It is a good practice because we don't know how the market will go in the future. When we ultimately trade according to our strategy, it will be on live data as it evolves, not on the historical price data used for backtesting [45].

Here's how out-of-sample testing works: First a backtest is performed on a given test period. Then the same backtest is run on a new test period – a different sample of data, hence the name. If the parameters or settings were over-optimized in the first backtest, it is unlikely that they will perform well in the second time period.

The Backtesting Engine is a software that does the backtesting ot the system. It takes the historical price data and trading strategies as inputs. The backtesting engine applies the trading strategies to the historical price data to get a series of hypothetical trades and records the results. The outputs of the backtesting engine are typically performance statistics. For example, its possible to tweak the parameters on just the right indicators to make over 1000% gains in backtesting. But when we run those same settings in another period, it might actually be losing. If it is custom fit to one set of data, it won't work as well in a different set of data [45].

6.2.1 *t*-test

In the ordinary t-test, a fundamental assumption is that the variance of the mean of a set of n independent observations is the original variance divided by n. But in the dualcorrelation problem, the returns themselves are serially correlated. If a set of observations is correlated this way, the variance of the mean does not drop by a factor of n. It drops more slowly. The result is that the t-score is inflated, resulting in excessive rejection of the null hypothesis. A crude fix is to estimate the serial correlation and apply a simple formula to correct the variance estimate.

6.3 Regression Testing

Regression test suites are necessary to ensure that changes made to the system after bug fixes or reimplementations have not corrupted the intended functionality. Common methods of regression testing include rerunning previously completed tests and checking whether the system behavior has changed. To perform such testing effectively regarding time, a systematic selection of an appropriate minimum of tests is needed.

6.4 Measurement Parameters

For the purpose of the use of a platform for high-frequency time series prediction, a few parameters are crucial. For the processing of high-frequency data, we need to be able to process these data and to have the response from the system in nearly real time. From this point of view, we found the following parameters for benchmark tests and their characteristics interesting as they are scalable and thus good to use for experimental measurements:

6.4.1 Latency

Latency is the lag between detection of two complex events in the set of triggering events sent to the CEP engine. In our setup, we note the time in milliseconds before sending each event. Upon matching a statement, the `updateListener` function would be invoked with the events. There we update the stats module with the current time – last event time.

6.4.2 Throughput

Throughput is the maximum number of events per second which the CEP engine can process without loss of data or without clogging the queues. The current setup uses a channel which blocks input on the application level if the channel buffers are full. So the client program will not be able to write data to the channel any faster than the server consumes it. At 100% CPU utilization, the throughput may decrease a little and the latency may increase.

6.4.3 CPU Utilization

This is the CPU Utilization for different kinds of CEP query over different event rates for a given pattern.

6.4.4 Memory Utilization

This is the memory profile for different kinds of CEP query over different event rates for a given pattern.

According to the Esper specification, Esper exceeds over 500 000 event/s on a dual CPU 2GHz Intel-based hardware, with engine latency below 3 microseconds average (below 10 microseconds with more than 99% predictability) on a VWAP benchmark with 1000 statements registered in the system – this tops at 70 Mbit/s at 85% CPU usage. Esper also demonstrates linear scalability from 100 000 to 500 000 event/s on this hardware, with consistent results across different statements [46].

6.5 CEP Benchmark Testing Frameworks

Currently, several solutions exist for measuring the performance of CEP platforms. Most of them started as a university project – FINCoS, BiCEP, CEPBen. We will describe the first of these, as it is more complex and flexible than others. The idea of all three is basically the same.

6.5.1 FINCOS

According to the [43], FINCoS is a set of benchmarking tools for load generation and performance measurement of various event processing systems. It allows the creation of synthetic workloads and enables the evaluation of candidate solutions, using the user's own datasets. An extensible set of adapters allows the framework to communicate with different CEP engines and its architecture permits the distribution of load generation across multiple nodes.

The FINCoS framework is composed of five main components:

- Drivers – simulate external event sources, submitting load to the system under test.
- System under test – tested CEP engine. The results produced by the system under test are received and stored in log files for subsequent answer validation and performance measurement.
- Sinks – receive the results produced by system under test.

- Adapters, Controller – communication with the CEP engine is carried out through an extensible set of adapters. Controller allows users to configure, execute, and monitor performance tests through GUI.

- Performance Monitor component – the results of performance tests can then be visualized both in real time and after test completion, using the Performance Monitor component.

The execution of drivers can be split into phases, each with its own workload characteristics. This is useful not only for breaking performance tests into well-described parts, but also for evaluating the ability of event processing platforms in adapting to changes in the load conditions. In addition, users can choose if events should be generated by the framework itself or read from files containing real-world event data.

Esper allows the balancing of loading of input data as each node is uniformly loaded. The workload can also be seamlessly scaled by simply adding more drivers and sinks to the configuration. The FINCoS framework supports direct communication with event-processing platforms through custom adapters.

6.6 Case Study

In order to prove the usefulness of the proposed approach, we applied the formalization of the business rules to the set of rules in the decision-making process. This section is devoted to an example of the use of the designed model in the execution of the Buy limit order. The Buy order is executed only if all conditions are met. The specific case is already described by Decision Table 4.1 described in Section 4.6 above.

Case description: Placement of the Buy (limit) order. Generally the Buy limit order is an order to purchase a security at or below a specified price. A Buy limit order allows traders and investors to buy a security with the restriction on the maximum price to be paid, or to sell a security with the restriction of the minimum price to be received. By using a Buy limit order, the investor is guaranteed to pay that price or lower. The order will only be executed at a specified limit price or better, but there is no guarantee that the order will be filled in the first place.

6.6.1 Construction of Knowledge Base

The construction of the rules set is described by matrix grammar $H = (G, M)$, $G = (N, T, P, S)$ for the table 4.1 is as follows.

We set actions from the table as nonterminal symbols.

- $N := (DONT\_BUY, BUY, S)$

Particular conditions create a set of terminals.

- $T := (valid\_symbol, valid\_quantity, sufficient\_funds, buy)$

The set of rules are created according to algorithm:

$P := (S \rightarrow < R1, C1 >< R1, C2 >< R1, C3 >, S \rightarrow < R2, C1 >< R2, C2 >< R2, C3 >,$
$S \rightarrow < R3, C1 >< R3, C2 >< R3, C3 >,$
$S \rightarrow < R4, C1 >< R4, C2 >< R4, C3 >,$

$< R1, C1 > \rightarrow DONT\_BUY valid\_symbol, < R2, C1 > \rightarrow DONT\_BUY valid\_symbol,$
$< R3, C1 > \rightarrow DONT\_BUY valid\_symbol, < R4, C1 > \rightarrow BUY valid_s ymbol,$
$BUY \rightarrow buy, DONT\_BUY \rightarrow \varepsilon,$
$< R1, C2 > \rightarrow valid\_quantity, < R1, C3 > \rightarrow sufficient\_funds,$
$< R2, C2 > \rightarrow valid\_quantity, < R2, C3 > \rightarrow sufficient\_funds,$
$< R3, C2 > \rightarrow valid\_quantity, < R3, C3 > \rightarrow sufficient\_funds,$
$< R4, C2 > \rightarrow valid\_quantity, < R4, C3 > \rightarrow sufficient\_funds)$

Where $< R_p, C_m >$ denotes nonterminals $< Rule_p, Condition_m >$ (according to the method in the formalization process described above), S denotes the starting nonterminal.

After the addition of nonterminals to the set of nonterminals N, N will look like:

$N := (DONT\_BUY, BUY, S, < R1, C1 >, < R1, C2 >, < R1, C3 >, < R2, C1 >, <$
$R2, C2 >,$
$< R2, C3 >, < R3 >, < R3, C2 >, < R3, C3 >, < R4, C1 >, < R4, C2 >,$
$< R4, C3 >)$

$M := [< R4, C1 > \rightarrow BUY valid\_symbol, BUY \rightarrow buy, < R4, C2 > \rightarrow valid\_quantity,$
$< R4, C3 > \rightarrow sufficient\_funds]$

Matrix M was determined on the basis of the execution of the Buy order, which is executed for the $Rule_4$ only and if only all the conditions are met.

6.7 Measurements

The decision-making module implemented and integrated into the Esper platform is based on EPL. The patterns in Esper take the form of SQL-like declarative rules that are given to the engine in the form of an uncompiled String, e.g.:

```
String epl = ‘‘select tick.price as tickPrice,
trade.price as tradePrice,
sum(tick.price) + sum(trade.price) as total
from pattern [every tick=StockTickEvent
or every trade=TradeEvent].win:time(30 sec)’’;
EPStatement statement = epService.getEPAdministrator().createEPL(epl);
```

Pattern syntax in Esper is done by using pattern statements. Pattern statements are created via the `EPAdministrator` interface. The EPAdministrator interface allows the creation of pattern statements in two ways:

- Pattern statements that want to make use of the EPL select clause, or
- Other EPL constructs use the createEPL method to create a statement that specifies one or more pattern expressions.

The use of the syntax is shown below.

```
EPAdministrator admin = EPServiceProviderManager.getDefaultProvider().
getEPAdministrator();
String eventName = ServiceMeasurement.class.getName();
EPStatement myTrigger = admin.createEPL(
"select * from pattern [" +  "every (spike=" + eventName + "(latency>20000)
or error=" + eventName + "(success=false))]");
```

All measurements were performed by using quad core 64-bit Operating System with Windows 7, 7-4600M CPU @ 290GHz 290 GHz Intel-based hardware with 16GB RAM. Information about EPL and pattern syntax is based on [46].

6.8 Test Cases

We measure the latency, throughput, CPU and memory utilization for our integrated solution. More detailed information about test cases is summarized in the following subsections.

6.8.1 Latency

Latency is a significant user metric in many real-time applications. Users are usually interested in quantiles of latency, such as worst case or 99th percentile. Measurement proved that the latency of the system was below 3 microseconds for 99%. This measurements significantly differs from the Esper indicated latency, but it may be caused by the complexity of the input data.

6.8.2 Throughput

Throughput is expressed in event/s. Experimental measurements proved that throughput ranged from 150 000 to 200 000 events processed per second. The measurement was performed no longer than 10 min after startup.

6.8.3 CPU Utilization

CPU utilization was measured within the range of 5-minute intervals. The applied load and the CPU usage correlated. The memory consumption was almost constant. The average count of threads which were processing at each moment was 16. The measurement was performed no longer than 10 min after startup. In Figure 6.1 the CPU utilization is captured within 5 minutes.

6.8.4 Memory Utilization

Memory utilization was measured within the range of 5-minute intervals. The average count of threads which were processing at each moment was 16. Memory heap utilization ranged between 100Mb and 350Mb of used memory. The measurement was performed no longer than 10 min after startup. In Figure 6.2 the CPU utilization is captured within 5 minutes.

6.8.5 Comparison to the Requirements for Real-time Processing

The rules for complex event stream processing summarized in Section 3.5 give clear guidelines for CEP. Most of the rules are satisfied by the use of the Esper platform.

- The first rule is completely fulfilled by the use of the complex Esper event platform: there is no need to store data during processing. The data are only persisted to a database after the analysis is complete. Esper provides real-time Big Data analytics for immediate insight, turning high-velocity log and other machine data into streaming operational intelligence. Instead of storing data, data arrive as real-time streams and are processed in-memory using continuous SQL-conforming queries. This allows for massively parallel streaming data processing.

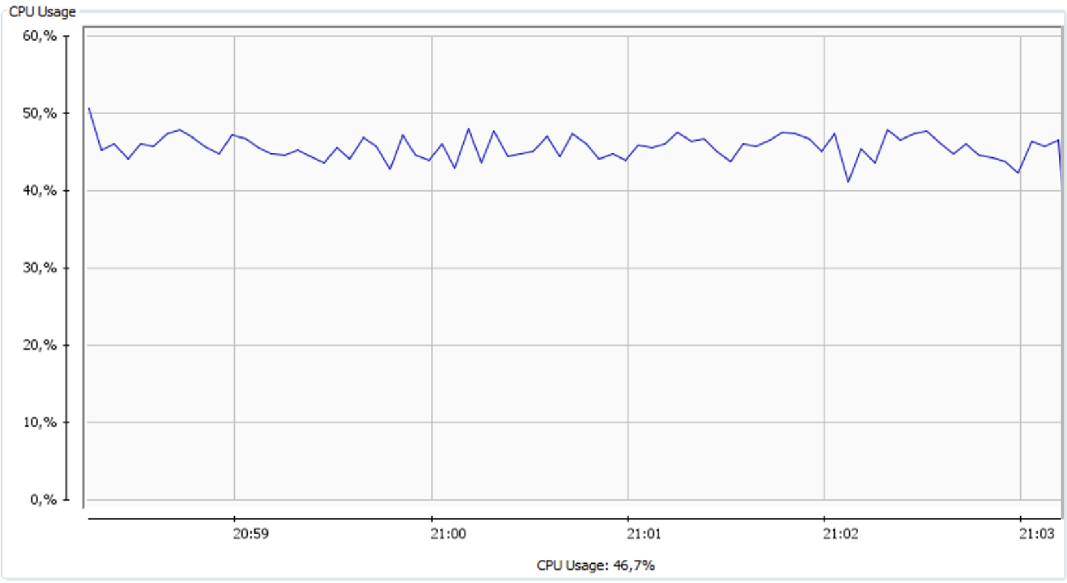


Figure 6.1: CPU Utilization. Source: author.

- The second rule is satisfied by the Esper EPL language. EPLs can be used to define event-processing agents. The EPL converges event stream processing (filtering, joins, aggregation) and complex event processing (causality) into one single language.
- The third and fourth rules, stream imperfections and ensuring predictable outputs are also implemented within the scope of the Esper tool. The data preprocessing phase contains the detection of outlier values and fault data. Ensuring predictable results is also important from the perspective of fault tolerance and recovery – i.e. the same input stream should yield the same outcome, regardless of the time of execution.
- The fifth rule states that the prediction runs over historical and live streaming data simultaneously. It is desirable to start computing on historical data and then to continue with the calculation on live data. This capability requires switching automatically from historical to live data, without the need for manual interventions. Again this requirement is fulfilled by chosen complex platform Esper.
- The sixth rule states that the data must be highly available and safe. These concerns were left out of the scope of our research. Also the architecture does not have any facilities to assist in the partitioning of the workload, as suggested by the seventh rule. Nevertheless, the CEP cluster provides a highly scalable platform which suits event processing networks well.
- The seventh requirement is to have the capability of distributing processing across multiple processors and machines to achieve incremental scalability. Ideally, the distribution should be automatic and transparent. The Esper component, EsperHA, is capable of distributed computing. In this thesis, we did not test this requirement, as we ran the solution on one single point.
- The last rule states that the CEP engine should be highly optimized. The components used in the Esper are known for their performance. At this point, optimization of

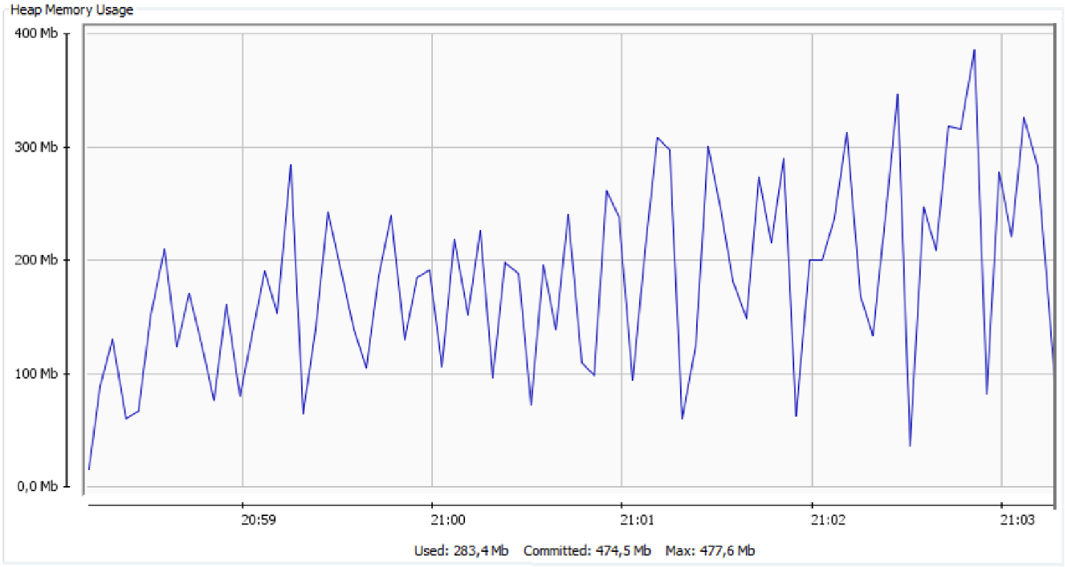


Figure 6.2: Memory Heap Utilization. Source: author.

the decision-making process in CEP Esper by using formalization of business rules is used. The results and comparison were measured on historical data.

This section was based on information from [4] and [50] and own experimental measurements and results which are summed up in [64].

Chapter 7

Conclusion

An increasing volume of information is being made available as online streams, and streams are expected to grow in importance in a variety of domains in the coming years (e.g. natural disaster response, surveillance, monitoring of criminal activity or military planning.

Currently, companies are collecting an increasing quantity of data, at an increasingly finer granularity. This stresses the need for new solutions and approaches to the processing of these data and also the predicting of their trends, in order to react to those data and make decisions based on them. The processing of such a quantity of data has become the basic requirement. For instance, in a variety of areas, such as algorithmic trading, identifying a trend or opportunity a few seconds or even milliseconds ahead of competition might mean the difference between success and complete failure

The objective of this thesis was to design and implement a new method for support of high-frequency data prediction. There are several areas of methods to assist in the prediction of data. We decided to use the existing platform for complex event processing and to integrate a new method for the description of the rules into the decision-making process. This process can be captured by using the decision support system. The result is the component of the decision support system – the set of business rules described by the model of formal grammar. The result was experimentally tested and measured over the set of historical data from the Forex financial market.

The main purpose for the implementation of our own decision-making system is to fully describe the business rules with a formal model. As a formal model, we chose the matrix grammar, as it allows the modeling of restrictions of actions upon the data and can partly simulate the parallel processing of actions within the scope of the business process. The implementation of this approach can be used for the formal verification of CEP systems. This area is still not fully explored.

7.1 Theoretical Contribution of the Thesis

In the scope ot this thesis was proposed a method for formal description of the ruleset used in decion-making process in complex event processing. This method have application in the proactive management of CEP.

7.2 Practical Contribution of the Thesis

The practical goal of this thesis was to implement a module for decision-making process. The resulting model was implemented into the existing CEP platform Esper. As already stated above, this method may contribute to the overall formal verification of CEP. Based on experimental results and measurement, the method help to speed up the processing of events.

7.3 Future Work

Despite the result achieved in this work, this topic still has many open areas for research. Future research involves some other possibilities of formalization of business rules by using some other tools, and the research of the possibilities of formally describing complex event processing. The current solution might be complemented by a graphical user interface (GUI) where user may update the set of business rules or correlate the parameters.

There currently exist many custom solutions which are often designed and used for the solution of one concrete problem, i.e. complex event processing. According to David Luckham, who is considered a pioneer in complex event processing, we are now in the era when CEP steps into more areas of business and there is a lack of standards and formalisms to describe and unite the approaches for the description of CEP platforms.

Bibliography

[1] Antonio Alegria, Complex Event Processing with Esper. 2016. [Online, visited September 2015].
Retrieved from: http://www.slideshare.net/antonio_alegria/complex-event-processing-with-esper-10122384

[2] Decision Management Solutions. 2016. [Online, visited 13.7.2016].
Retrieved from: http://www.decisionmanagementsolutions.com/

[3] DMN 1.1 Tutorial, Get Started with Decision Management using DMN. 2016. [Online, visited 26.8.2016].
Retrieved from: https://camunda.org/dmn/tutorial/#bpmn

[4] EsperTech, Event Series Intelligence. 2016. [Online, visited 26.8.2016].
Retrieved from: http://www.espertech.com/esper/release-5.4.0/esper-reference/pdf/esper_reference.pdf

[5] Ibm.com. 2016. [Online, visited 13.7.2016].
Retrieved from: http://www.ibm.com/developerworks/bpm/bpmjournal/1206_boyer/1206_boyer.html

[6] Ifrs.org. 2016. [Online, visited 10.7.2016].
Retrieved from: http://www.ifrs.org/Pages/default.aspx

[7] Openrules.com. 2016. [Online, visited 13.7.2016].
Retrieved from: http://openrules.com/.

[8] Thecepblog.com. 2016. [Online, visited 11.7.2016].
Retrieved from:
http://www.thecepblog.com/what-is-complex-event-processing/

[9] Tibco.com. 2016. [Online, visited 12.7.2016].
Retrieved from:
http://www.tibco.com/products/event-processing/complex-event-processing

[10] Tick Data: Historical Forex, Options, Stock and Futures Data. 2016. [Online, visited 12.6.2016].
Retrieved from: https://www.tickdata.com/product/historical-forex-data/

[11] What Is Forex? 2016. [Online, visited 10.7.2016].
Retrieved from: http://amalfx.com/forex.html

[12] Aalto, A.: *Scalability of Complex Event Processing as a part of a distributed Enterprise Service Bus*. PhD. Thesis. Aalto University School of Science. 2012.

[13] Akerkar, R.: *Big Data Computing*. Chapman & Hall/CRC. 2013. ISBN 1466578378, 9781466578371.

[14] Aldridge, I.: *High-frequency trading*. Wiley. 2010. ISBN 978-1-118-34350-0.

[15] Arango, M.: Mobile QoS Management Using Complex Event Processing: (Industry Article). In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*. DEBS '13. New York, NY, USA: ACM. 2013. ISBN 978-1-4503-1758-0. pp. 115–122. doi:10.1145/2488222.2488277.
Retrieved from: http://doi.acm.org/10.1145/2488222.2488277

[16] Arlt, A. M., Josef: *Ekonomické časové řady*. Grada. 2007. ISBN 978-80-247-6360-6.

[17] babypips.com: *What is Forex: Market Size And Liquidity [online]*. [Online; navštíveno 29.8.2016].
Retrieved from: http://www.babypips.com/school/preschool/what-is-forex/market-size-and-liquidity.html

[18] Bauwens, L.: *High frequency financial econometrics*. Physica-Verl.. 2008.

[19] Box, G. E.; Jenkins, G. M.; Reinsel, G. C.; et al.: *Time series analysis*. John Wiley & Sons, Inc.. 2016. ISBN 978-1-118-67502-1.

[20] Brooks, C.: *Introductory econometrics for finance*. Cambridge University Press. 2010.

[21] Copeland, L.: *A practitioner's guide to software test design*. Artech house. 2004.

[22] Dacorogna, M. M.: *An introduction to high-frequency finance*. Acad. Press. 2001.

[23] Daníelsson, J.: *Financial Risk Forecasting: The Theory and Practice of Forecasting Market Risk, with Implementation in R and Matlab (Wiley Finance Series)*. John Wiley and Sons Incorporated. 2011.

[24] Daníílsson, J.: *Financial Risk Forecasting: The Theory and Practice of Forecasting Market Risk, with Implementation in R and Matlab (Wiley Finance Series)*. John Wiley & Sons Incorporated. 2011.

[25] Debevoise, T.: *Business process management with a business rules approach*. Business Knowledge Architects. 2005.

[26] Dunkel, J.; Bruns, R.; Pawlowski, O.: Complex event processing in sensor-based decision support systems. *Nag, B.(Hg.) Intelligent Systems in Operations*. 2010: pp. 64–79.

[27] Durbin, M.: *All about high-frequency trading*. McGraw-Hill. 2010.

[28] Etzion, P., OpherNiblett: *Event processing in action*. Manning. 2011.

[29] Franke, J.; Härdle, W.; Hafner, C.: *Statistics of financial markets*. Springer. 2011.

[30] Hall, J. S. A. B. C. T., David Llinas: *Handbook of multisensor data fusion*. CRC Press. 2001.

[31] Hypský, R.; Zámečníková, E.; Kreslíková, J.: Formal Definition of Business Rules by Grammar Systems. *International Journal of Advancements in Communication Technologies*. vol. 2, no. 1. 2015.

[32] Jao, C. S.: *Efficient decision support systems*. 2011.

[33] Ji, Y.; Heinze, T.; Jerzak, Z.: HUGO: Real-time Analysis of Component Interactions in High-tech Manufacturing Equipment (Industry Article). In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*. DEBS '13. New York, NY, USA: ACM. 2013. ISBN 978-1-4503-1758-0. pp. 87–96. doi:10.1145/2488222.2488272.
Retrieved from: http://doi.acm.org/10.1145/2488222.2488272

[34] Jürgen, D.; Fernández, A.; Ortiz, R.; et al.: *Injecting Semantics into Event-Driven Architectures*. Springer. 2009.

[35] Kellner, L., I.Fiege: Viewpoints in complex event processing: Industrial experience report. *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems - DEBS '09*. 2009: pp. 91–98. doi:10.1145/1619258.1619260. [Online, retrieved 2016-7-11.

[36] Kirchgässner, J., GebhardWolters: *Introduction to modern time series analysis*. Springer. 2007.

[37] Kroha, P.; Friedrich, M.: *Comparison of Genetic Algorithms for Trading Strategies*. Cham: Springer International Publishing. 2014. ISBN 978-3-319-04298-5. pp. 383–394. doi:10.1007/978-3-319-04298-5_34.
Retrieved from: http://dx.doi.org/10.1007/978-3-319-04298-5_34

[38] Lai, H., T. Xing: *Statistical models and methods for financial markets*. Springer. 2008. ISBN 978-0-387-77827-3.

[39] Lovrencic, S.; Rabuzin, K.; Picek, R.: Formal Modelling of Business Rules: What Kind of Tool to Use? *Journal of information and organizational sciences*. vol. 30, no. 2. 2006.

[40] Luckham, D. C.: *The power of events*. Addison-Wesley. 2002.

[41] Luckham, D. C.: *Event Processing For Business*. Wiley. 2012.

[42] Martinsky, O.: *Intelligent trading systems*. Harriman House. 2010.

[43] Mendes, M. R.; Bizarro, P.; Marques, P.: FINCoS: Benchmark Tools for Event Processing Systems. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ICPE '13. New York, NY, USA: ACM. 2013. ISBN 978-1-4503-1636-1. pp. 431–432. doi:10.1145/2479871.2479941.
Retrieved from: http://doi.acm.org/10.1145/2479871.2479941

[44] Nag, B.: *Intelligent systems in operations*. Business Science Reference. 2010.
Retrieved from: http://www.irma-international.org/viewtitle/42655/

[45] Narang, R. K.: *Inside the Black Box: The Simple Truth About Quantitative Trading*. Wiley. 2013. ISBN 978-1-118-36241-9.

[46] Pestana, G.; Heuchler, S.; Casaca, A.; et al.: Complex Event Processing for Decision Support in an Airport Environment. *International Journal on Advances in Software Volume 6, Number 3 & 4, 2013*. 2013.

[47] Power, D. J.: Resources for Students Index at DSSResources.COM. 2016. [Online, visited -7-13.
Retrieved from: http://dssresources.com/dssbook/

[48] Ross, R. G.: *Business rule concepts*. Business Rule Solutions. 2009.

[49] Rozenberg, G.; Salomaa, A.: *Handbook of formal languages*. Springer. 1997.

[50] Stonebraker, M.; Çetintemel, U.; Zdonik, S.: The 8 requirements of real-time stream processing. *ACM SIGMOD Record*. vol. 34, no. 4. 2005: pp. 42–47.
doi:10.1145/1107499.1107504.
Retrieved from: http://cs.brown.edu/~ugur/8rulesSigRec.pdf

[51] Taylor, J.: *Decision management systems*. IBM Press/Pearson Education. 2012.

[52] Teymourian, K.; Rohde, M.; Paschke, A.: Knowledge-based Processing of Complex Stock Market Events. In *Proceedings of the 15th International Conference on Extending Database Technology*. EDBT '12. New York, NY, USA: ACM. 2012. ISBN 978-1-4503-0790-1. pp. 594–597. doi:10.1145/2247596.2247674.
Retrieved from: http://doi.acm.org/10.1145/2247596.2247674

[53] Toshchakov, I., IgorToshchakov: *Forex*. Piter. 2010.

[54] Tovarňák, D.; Nguyen, F.; Pitner, T.: Distributed Event-Driven Model for Intelligent Monitoring of Cloud Datacenters. Springer International Publishing Switzerland. 2014. ISBN 978-3-319-01570-5.

[55] Tsay, R. S.: *Analysis of financial time series*. Wiley, New York. 2002.

[56] Turban, E., EfraimTurban: *Decision support and business intelligence systems*. Pearson Prentice Hall. 2007.

[57] Von Halle, B.; Goldberg, L.; Zachman, J. A.: *The business rule revolution*. HappyAbout.info. 2006.

[58] Zámečníková, E.: Design Of Autonomous Algorithmic Models For Time Series Prediction. In *STUDENT EEICT 2014*. Volume 3. Brno University of Technology. 2014. ISBN 978-80-214-4924-4. pp. 279–283.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10584

[59] Zámečníková, E.; Kreslíková, J.: Design of Adaptive Business Rules Model for High Frequency Data Processing. In *ISAT Monograph Series*. Wroclaw University of Technology. 2014. ISBN 978-83-7493-346-9. BEST CONTRIBUTION AWARD. pp. 1–10.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10669

[60] Zámečníková, E.; Kreslíková, J.: Time Series Prediction Based on Event Driven Business Process Management. *International Journal of Computational Engineering Research (IJCER)*. vol. 4, no. 4. 2014: pp. 1–6. ISSN 2250-3005.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10599

[61] Zámečníková, E.; Kreslíková, J.: Comparison of Platforms For High Frequency Data Processing. In *2015 IEEE 13th International Scientific Conference on Informatics*. The University of Technology Košice. 2015. ISBN 978-1-4673-9867-1. pp. 296–301.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10964

[62] Zámečníková, E.; Kreslíková, J.: Formalization of Business Rules in Decision Making Process. In *Work in Progress Session SEAA 2015 41st Euromicro Conference on Software Engineering and Advanced Application*. Johannes Kepler University Linz. 2015. ISBN 978-3-902457-44-8. pp. 13–14.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=10829

[63] Zámečníková, E.; Kreslíková, J.: Business Rules Definition for Decision Support System Using Matrix Grammar. *Acta Informatica Pragensia*. vol. 5, no. 1. 2016: pp. 72–81. ISSN 1805-4951.
Retrieved from: http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11122

[64] Zámečníková, E.; Kreslíková, J.: Performance Measurement of Complex Event Platforms. *Journal of Information and Organizational Sciences*. vol. 40, no. 2. 2016. ISSN 1846-9418. IN REVIEW PROCESS.

Appendices

List of Appendices

Appendix A

CD Content

- The electronic version of this PhD Thesis.
- Source codes of CEP Esper engine with integrated solution.
- Copies of publications.

Appendix B

What is traded on FOREX?

Symbol	Country	Currency
USD	United States	Dollar
EUR	Euro zone members	Euro
JPY	Japan	Yen
GBP	Great Britain	Pound
CHF	Switzerland	Franc
CAD	Canada	Dollar
AUD	Australia	Dollar
NZD	New Zealand	Dollar

Table B.1: Major currencies traded on FOREX – based on [17].

Currency symbols always have three letters, where the first two letters identify the name of the country and the third letter identifies the name of that country's currency (Table B.1).

Appendix C

Market size and liquidity

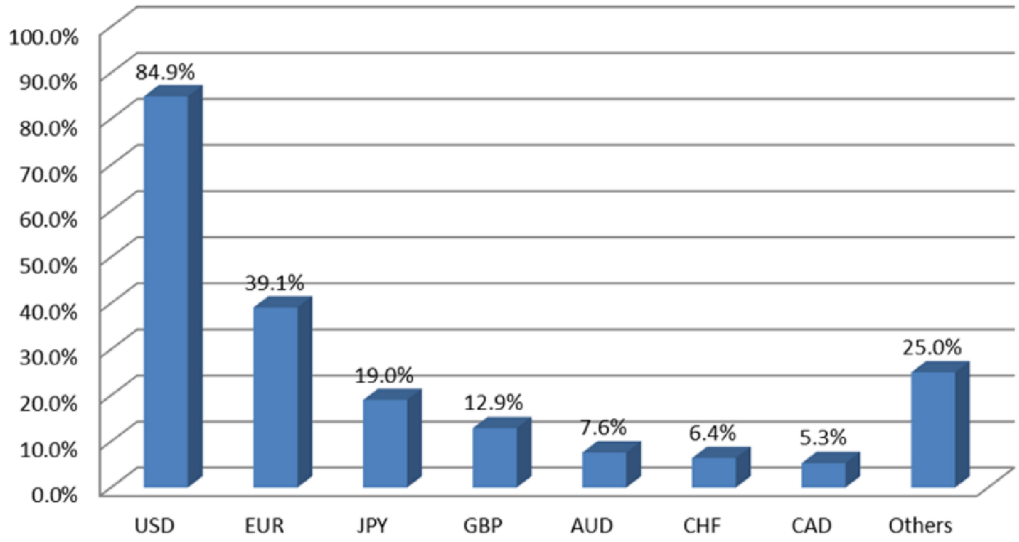


Figure C.1: Currency distribution in the FOREX market. Source: [17].

Because two currencies are involved in each transaction, the sum of the percentage shares of individual currencies totals 200% instead of 100%.

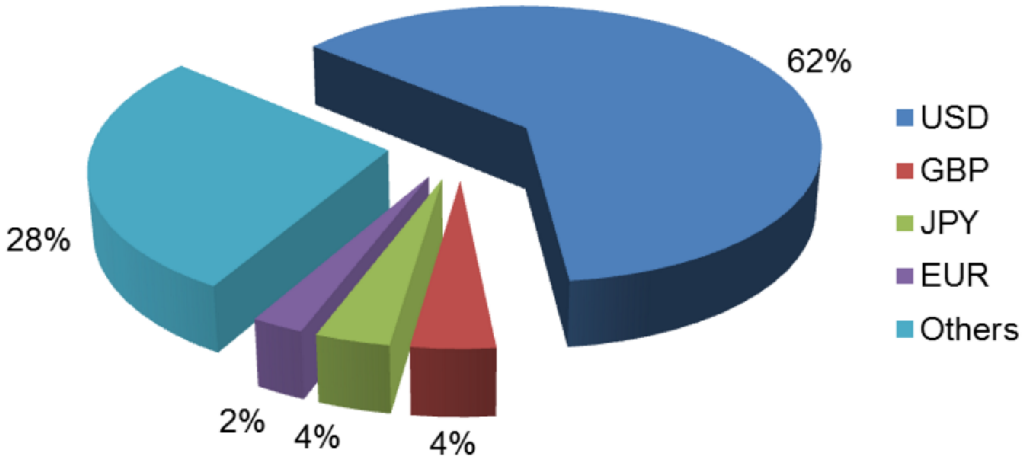


Figure C.2: Currency composition of world FOREX reserves. Source: [17].

There are significant reasons why the U.S. dollar (Figure C.1 and C.2) plays a central role in the FOREX market:

- The United States economy is the largest economy in the world.
- The U.S. dollar is the reserve currency of the world.
- The United States has the largest and most liquid financial markets in the world.
- The United States has a super stable political system.
- The United States is the world's sole military superpower.
- The U.S. dollar is the medium of exchange for many cross-border transactions. For example, oil is priced in U.S. dollars.