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.

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 [27]. In his next monography about complex events, *Event Processing For Business* [28], 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 [28]. 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.

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 [11], [12], [23] and [37], 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 [40].

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 [11], 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").*

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 [18].

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 [42]. 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.

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 [11].

This section was based on [9], [17] and [25].

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 [29].

According to David Luckham [27], 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 [14], [27].

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.

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.

3.1 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 [27]. 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 [27] 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).

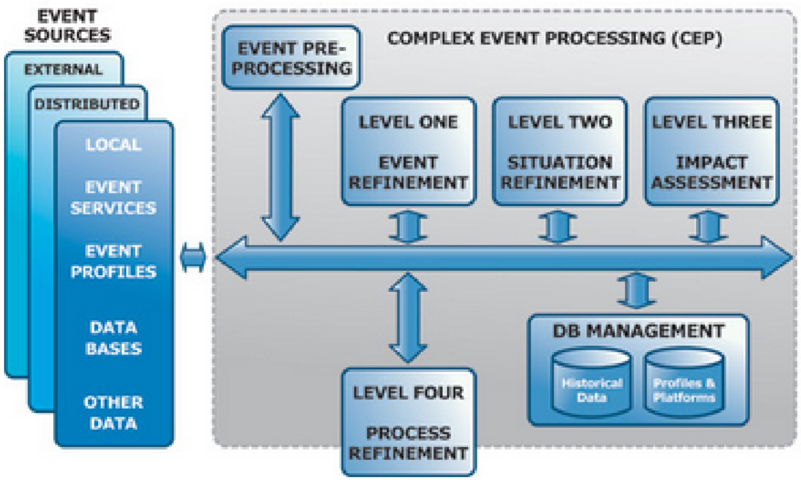

Figure 3.1: Complex event processing reference architecture – adapted from [5].

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 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 [42].

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 – aggregation-oriented CEP or 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.

3.2 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 [27]. 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 [16], [29].

Unfortunately, event-driven architectures have not yet the maturity of well-established software architectures: there is still a lack of methodologies, models and standards [16].

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.

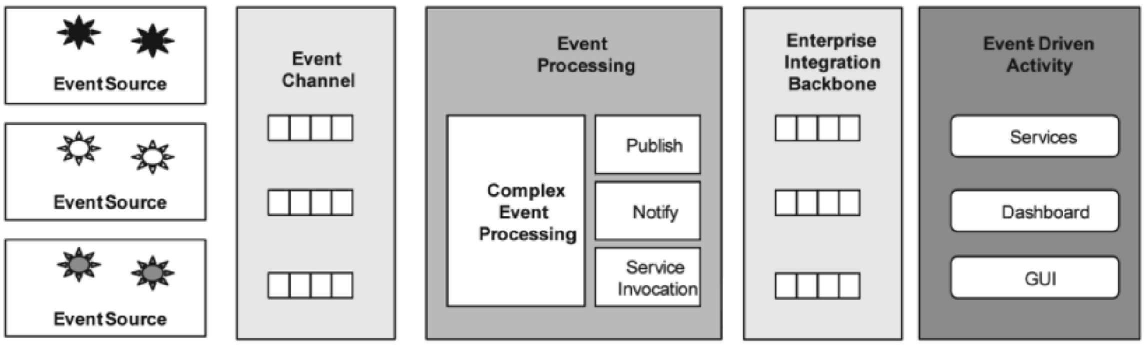

Figure 3.2: Flow of events in event-driven architecture – adapted from [29].

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.

3.3 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. 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.

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 [41].

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 [8].

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 [6].

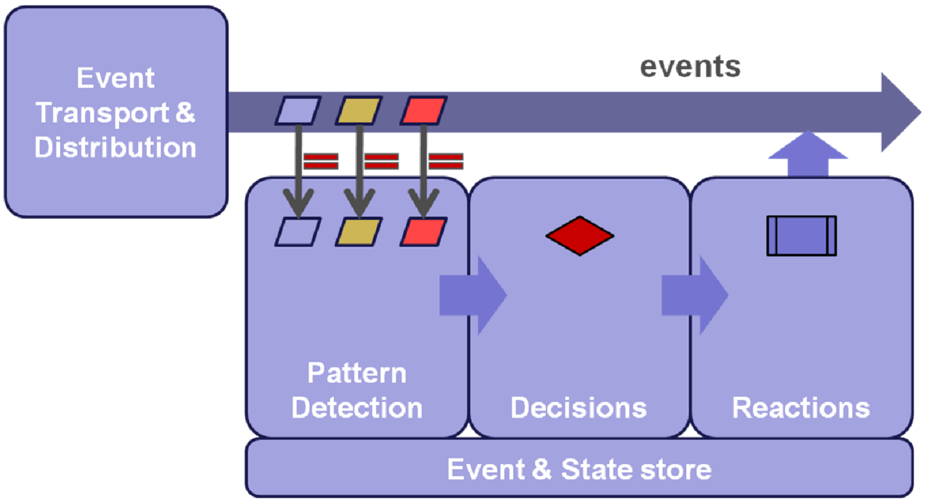

Figure 4.1: Decision making process schema – adapted from [6].

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 [29]. Workflows are set up according to the characteristics of input data.

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.

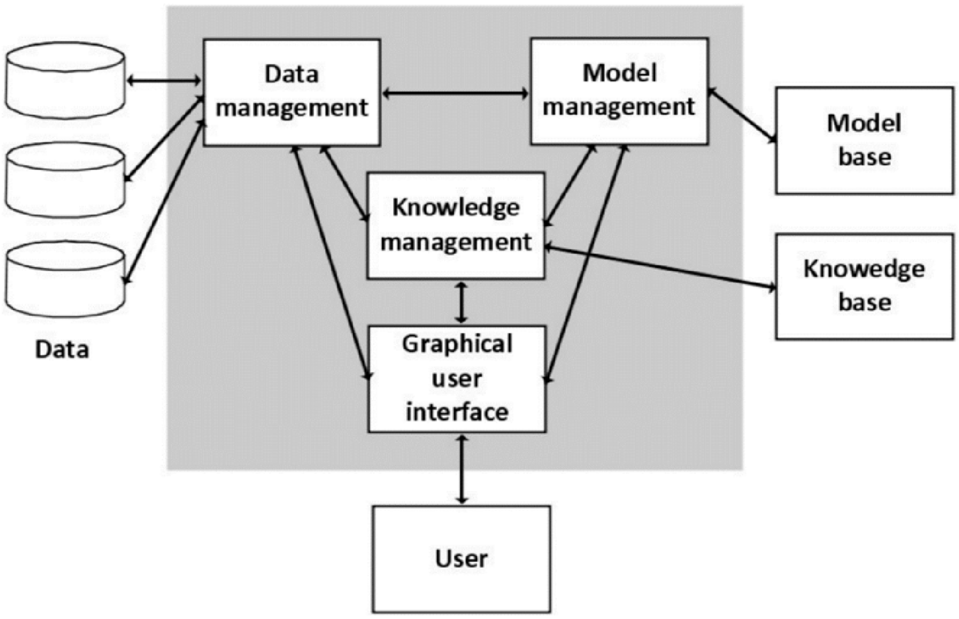

Figure 4.2: Schema of a Decision Support System – based on [20].

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 [34], 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 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], [15], [31] and [34].

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 [31] or Turban [38], 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 [20] and [31]. 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) [29].

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 [31]. 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 [39] 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 [6].

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.

The basic principles about this topisc are also summed up in [43].

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. 4.3.

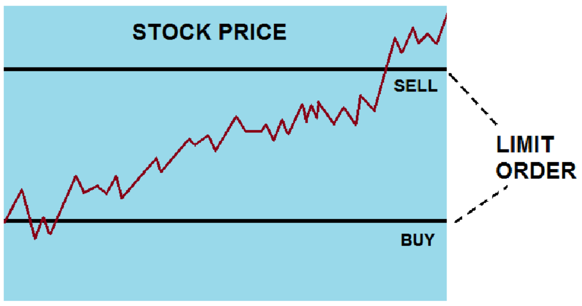

Figure 4.3: Example of Limit Order. Source: author.

4.4.2 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.

4.5 Decision Tables

Decision tables 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 [13].

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.

4.5.1 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 [4]. 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 [16].

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 [10].

Other application of formalization is introduced in [22] 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 [8] and [36] 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 [24] 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 [21] 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 [26] 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 [35]. 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 [19], 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 [32]. The definition of matrix grammar follows.

Definition 2 *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.

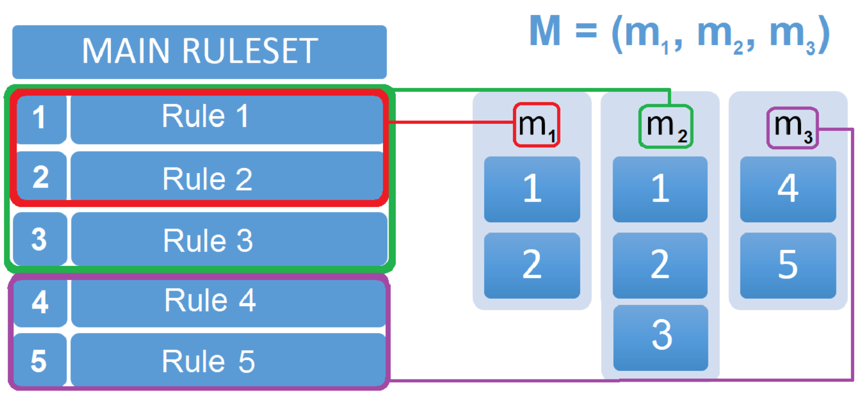

Figure 5.1: Sample ruleset of decision support component and matrices mapping. Source: author.

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;
$N := \{Action_1, Action_2, \dots, Action_n\}$;
$T := \{condition_1, condition_2, \dots, condition_m, action_1, action_2, \dots, action_n\}$;
$P := N \times (N \cup T)^*$
foreach $Rule_p$, where $Rule_p \in \{Rule_1, Rule_2, \dots, Rule_p\}$ from the decision table
 consider all suffice conditions, the set $\{Condition_1, Condition_2, \dots, Condition_m\}$
 do
     1. add rule $p, p \in P : S \to < Rule_1, Condition_1 >< Rule_1, Condition_2 > \cdots < Rule_1, Condition_m >$;
     2. add rule $p, p \in P : S \to < Rule_2, Condition_1 >< Rule_2, Condition_2 > \cdots < Rule_2, Condition_m >$;
     3. ...;
     4. add rule $p, p \in P : S \to < Rule_p, Condition_1 >< Rule_p, Condition_2 > \cdots < Rule_p, Condition_m >$;
     5. add $< Rule_p, Condition_m >$ to $N$; $m, p$ are positive integers.
end
foreach $< Rule_p, Condition_1 >$ do
    add rule:
     • $< Rule_p, Condition_1 > \to Action_n condition_1$
     • $Action_n \to action_1 action_2 \dots action_n$, 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:
     • $< Rule_p, Condition_m > \to condition_m$
end
$S := S$; (~waiting state);
$M := \{m_1, m_2, \dots, m_p\}$, where $m_p = [< Rule_p, Condition_1 >$
 $Action_n condition_1, < Rule_p, Condition_m > \to condition_m$ 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 [43], [44] and [45].

5.2.3 Quote Data

As sample data, we took historical financial high-frequency data from Forex. Data are available at [7] 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
```

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 [7].

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 [3] and the web pages of this tool [30].

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.

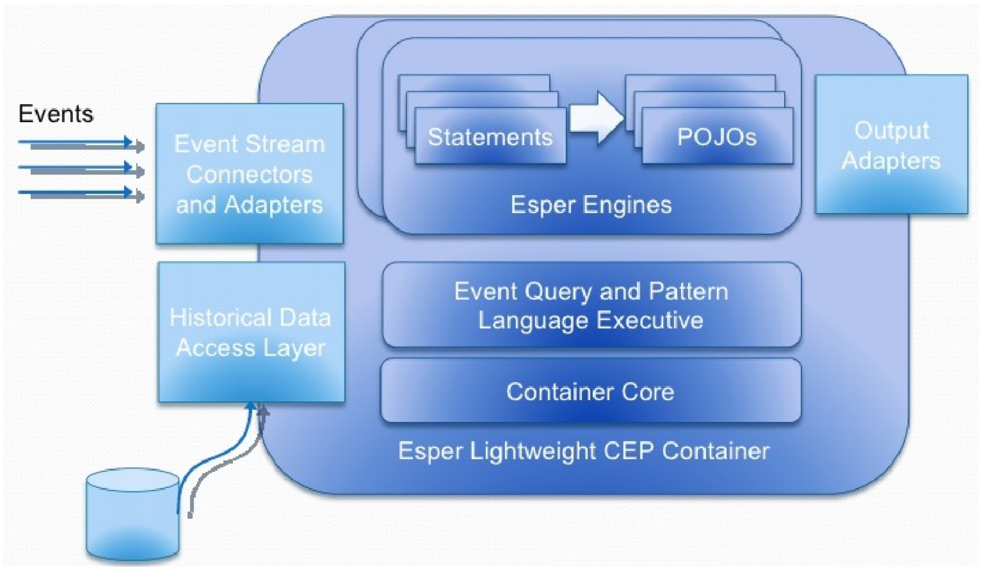
Events
Event Stream Connectors and Adapters
Statements
POJOs
Esper Engines
Output Adapters
Historical Data Access Layer
Event Query and Pattern Language Executive
Container Core
Esper Lightweight CEP Container

Figure 5.2: Core of Esper CEP platform – adapted from [28].

Figure 5.2 shows a core of the Esper engine. The Esper engine is based on the use of state machine technology. 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 [30].

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.

5.4 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 5.3:

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.

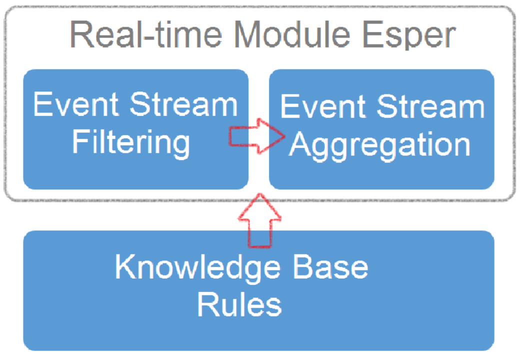

Figure 5.3: Schema of implemented DSS. Source:author.

5.4.1 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.

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 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:

6.1.1 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.

6.1.2 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.

6.1.3 CPU Utilization

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

6.1.4 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 [30].

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.

6.1.5 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:

$P := (S \to < R1, C1 >< R1, C2 >< R1, C3 >, S \to < R2, C1 >< R2, C2 >< R2, C3 >,$
$S \to < R3, C1 >< R3, C2 >< R3, C3 >,$
$S \to < R4, C1 >< R4, C2 >< R4, C3 >,$
$< R1, C1 > \to DONT\_BUY valid\_symbol, < R2, C1 > \to DONT\_BUY valid\_symbol,$
$< R3, C1 > \to DONT\_BUY valid\_symbol, < R4, C1 > \to BUY valid_s ymbol,$
$BUY \to buy, DONT\_BUY \to \varepsilon,$
$< R1, C2 > \to valid\_quantity, < R1, C3 > \to sufficient\_funds,$
$< R2, C2 > \to valid\_quantity, < R2, C3 > \to sufficient\_funds,$
$< R3, C2 > \to valid\_quantity, < R3, C3 > \to sufficient\_funds,$
$< R4, C2 > \to valid\_quantity, < R4, C3 > \to sufficient\_funds)$

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:

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

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

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.

6.2 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 [30].

6.3 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.

6.3.1 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.

6.3.2 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.

6.3.3 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.

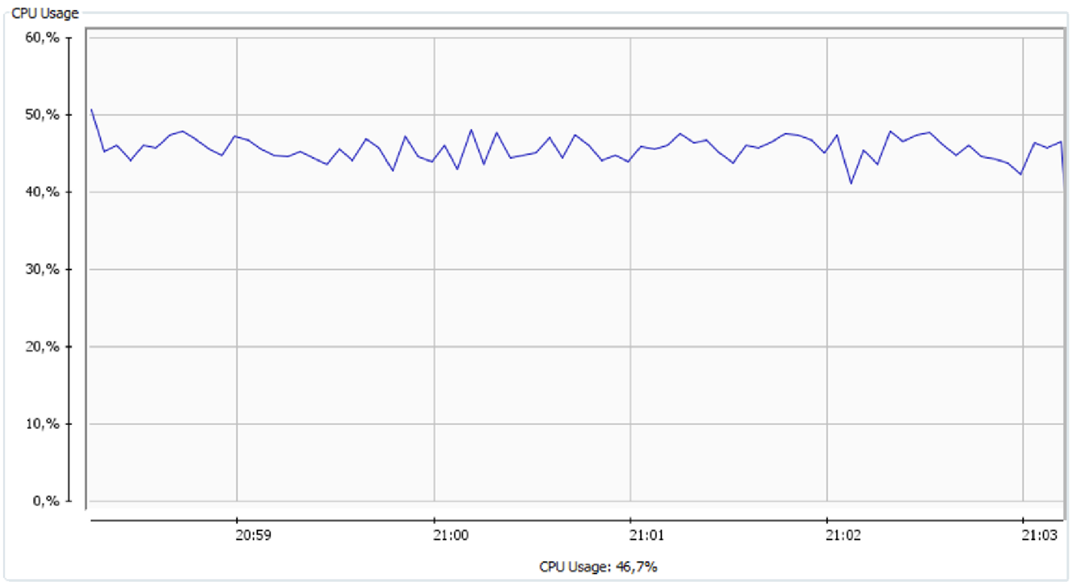

Figure 6.1: CPU Utilization. Source: author.

6.3.4 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.

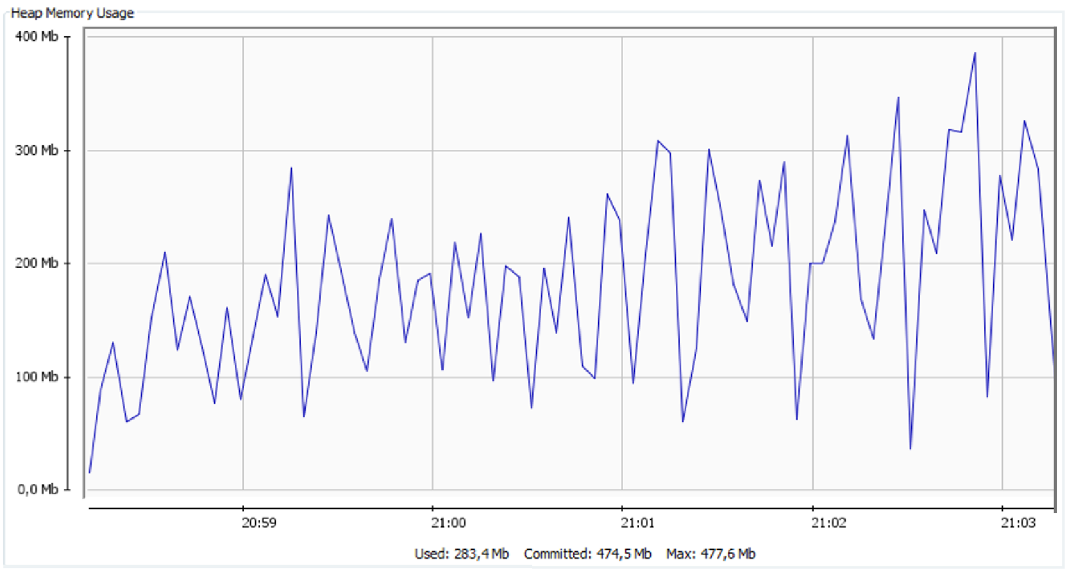

Figure 6.2: Memory Heap Utilization. Source: author.

This section was based on information from [3] and [33] and own experimental measurements and results which are summed up in [46].

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] 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

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

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

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

[7] 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/

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

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

[10] 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

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

[12] 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.

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

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

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

[16] 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.

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

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

[19] 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.

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

[21] 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

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

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

[24] 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

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

[26] 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.

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

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

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

[30] 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.

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

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

[33] 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

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

[35] 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

[36] 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.

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

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

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

[40] 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

[41] 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

[42] 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

[43] 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

[44] 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

[45] 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

[46] 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.

Curriculum Vitae

Eva Zamečníková, 26.6.1985

CONTACT INFORMATION

Chaloupky 204
Bystřice pod Hostýnem
768 61
Czech Republic

Phone: +420 777 899 032
E-mail: evazamek@gmail.com

EDUCATION

Brno University of Technology
Faculty of Information Technology BUT

Ph.D. Degree Study **2009 till now**

- Study programme: Computer Science and Engineering
- Thesis topic: Formal Model of Decision Making Process for High-Frequency Data Processing
- expected graduating date: 2016

Master's Degree Study **2007 - 2009**

- Study programme: Information Systems
- Topic of Master's Project: *Syntactic Analysis Based On Pair Automata* The design and implementation of a special synactic analysis method in C++ language.

Bachelor's Degree **2004 - 2007**

- Study programme: Information Technology

Grammar school: Gymnázium Ladislava Jaroše, Holešov **2000 - 2004**

PROFESSIONAL EXPERIENCE

HOME CREDIT INTERNATIONAL, a.s.

- Projects:

LAP, embedit.cz **May 2014 - now**

- Ligh Approval Process, Developing of application for approving of customers contracts.
- J2EE developer, development of front-end, back-end application and bussines logic.

UNICORN SYSTEMS, a.s.

- Projects:

LAP, embedit.cz **March 2013 - April 2014**

- Developing of application for approving of customers contracts.
- J2EE developer, development of front-end, back-end application and bussines logic.

CTDS (Common Tool for Data Exchange and Security Assessment) **October 2012 - February 2013**

- Project for Industry Solutions customer.
- J2EE developer: development of front-end, back-end application, application logic and business logic.

Tatra Banka, TBRS **May 2012 - September 2012**

- Project for Slovak Bank Tatra Bank. Developing of internal portal for customer service.
- J2EE developer: development of front-end, back-end application and aplication logic.

MATCOMP

Development of web applications **March 2011 - March 2013**

- Development of interactive applications in Flash, AS3.

COMPUTER SKILLS	**Programming Languages** Java, C/C++, PHP
	Databases MySQL, Oracle
	Web HTML, JavaScript, CSS
	Development Tools IntelliJ IDEA, Eclipse
	Others LaTeX, XML, J2EE, Hibernate, Wicket, Spring, Swing, SEAM, Log4j, jBPM, UML, Craft.CASE (process modelling), Flash AS3, Git, SVN, XQuery
LANGUAGE SKILLS	**English** advanced proficiency
	German elementary proficiency
	Spanish elementary proficiency
	Czech native speaker
OTHERS, CERTIFICATES	• driving licence category B
	• FCE, B2
	• Java Essentials
INTERESTS	Education - new technologies, music, travelling