MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Analysis and Design of Deployment of Complex Event Processing

DIPLOMA THESIS

Martin Rusyniak

Brno, Spring 2015

Declaration

Hereby, I declare that this diploma thesis is my original work created solely by me, the author. All of the sources used for its elaboration are cited and stated in the Literature.

Martin Rusyniak

Advisor: RNDr. Filip Nguyen

Acknowledgement

I am very thankful for support provided to me by RNDr. Filip Nguyen, who helped and inspired me during hard-times of elaboration of this thesis.

I also thank to multiple employees of ACME Company, as without their assistance, this thesis could not be created. Notably František, Michael and Miroslav.

Abstract

This diploma thesis focuses on application of Complex event processing (CEP) in IT business.

Events, as specific occurrences, are valuable source of information about what is happening. Ability to catch, process and evaluate them, can provide information of high business value.

The thesis explains basic theory and usual applications of CEP. Afterwards, it analyses company's environment and requirements for CEP system. The ultimate outcome of this diploma thesis is the architecture and examples of rules for complex event processing.

Keywords

cep, complex event processing, event, real time business intelligence, real time processing, business rule, soa, enterprise

Table of Contents

Introduction

"Complex event processing (CEP) is a set of techniques and tools to help us understand and control event-driven information systems." [6]

Events, as specific occurrences, are valuable source information about what is happening. Ability to catch, process and evaluate them, can provide information of high business value. Complex event processing therefore opens up new vistas of business intelligence and allows us to distinguish, what is important, in near real-time.

This diploma thesis focuses on analysis and design of deployment of Complex event processing system for obtaining results of business value in ACME Company.

Theory related to Complex event processing is placed in chapter 1, containing basic terms and features of this method, and chapter 2, containing guidance on building and operation of such systems. Chapter 3 then describes benefits of and use cases for implementation of this approach.

Chapter 4 speaks of the environment, into which a solution for Complex event processing is going to be deployed. Chapter 5 presents the business analysis for this deployment, describing mainly stakeholders, requirements and data involved. This continues to chapter 6 containing the ultimate outcome of this diploma thesis, the architecture and examples of rules for complex event processing. Also, analytical and design chapters describe sources of events and their interconnection.

The last chapter of this diploma thesis, Conclusion, summarises this work and outlines possible areas of interest and steps for the future.

The main added value of this thesis is in analysis and design parts, which are ground phases setting fundaments for any project. Complex event processing is a powerful method with potential to be deployed in multiple domains for various applications, many of which are yet to come. Therefore, this thesis can also be used as a guide for analysts in other organisations analysing and designing deployment of CEP.

1. Theoretical foundations

This chapter contains basic theory, terms and explanations, we will be later using and building upon. It is based mostly on publications [3], [6] and [5] and provides consistent overview of the most important terms and features of this method. Though, some of the information is self-explanatory and understandable right from its name.

1.0.1 Event

According to the dictionary [25], an event is something that happens. We can broaden this term and include also *virtual* events. These are currently not real, but they happen as a consequence of some action. And then, they would happen. [6][5]

An event is not just a message, though it may have form of a message. For us, this term presents whole object specifying activity and containing event-specific data, e.g. originator, timestamp, duration, significance, other values describing event and relationships to other events. These relationships are important for any further processing and can be defined by time, causality or aggregation, and similar. [6]

1.1 Causality and Layers

1.1.1 Event Causality

In order to understand events, we must know what *caused* them and we need to have this information *at the time of given event*. Though, we know that both physical and electronic worlds are often too complex to thoroughly understand and therefore stating cause for every event is not always possible. [6]

Set of identifiers of events that are a cause for some other event can be called a *causal vector.* [6]

1.1.2 Aggregation

Event aggregation means making sense out of huge number of events that occur. The events that compose the whole are called *members* and we use term *complex event* for the entire aggregation. Causality relationships between events are both transitive and asymmetric, therefore we have a partial ordering on sets of events. Together, they are partially ordered sets, also called posets, which can be easily drawn as a *directed acyclic graph* (DAG). [6]

Term complex event was created and explained in [6] in order to have a better differentiation from simple, atomic, event. With publishing of [5], from the same author, the stress on differentiation of these terms is weaker. From our point of view, term "complex event" is more useful, when speaking of business level, while using more general term "event" suits better for lower levels.

1.1.3 Layers and abstraction hierarchy

To have a better understanding of events happening in our system, we need to differentiate them according to in which logical and hierarchical level do they

occur. Therefore, we divide events into *layers,* or also *levels*. Their meaning is to differentiate events and their impact from business, as well as system, point-of-view. [6]

For building layers, we can use usual terminology *low-level* and *high-level event.* But usually, we will use more levels to have better data granularity and understanding. In any enterprise, decision makers need to have aggregated data in order to make meaningful decisions. That's where CEP comes handy.

More information on abstraction hierarchy is available in the Appendix.

1.2 Monitoring targets

With events flowing into a machine processing them, we require few features so that the processing is value-adding. These features are to [5]:

- Monitor and analyse events at every architectural level of the system
- Be able to detect complex events composed from events widely distributed in their time and location
- Trace causal relationships between any events in real time
- Take action after detection of complex event of our interest
- Design systems to exploit benefits provided by CEP via proper monitoring, viewing and action taking

1.2.1 Event Stream

There are CEP solutions processing linear sequence of flowing events, *an event stream*. These solutions do not possess information about relations between inflowing events and are therefore easier to implement. These solutions may be less effective and may not uncover some of the important consequences, since the information about relationship is missing. [5]

1.2.2 Event Cloud

We speak of *event cloud*, when information about relationships is available and, at the same time, more streams of events are processed. Having an information about timing and causality allows event mining with probability of obtaining better outcomes. Also, many industrial problems cannot be taken as one stream only and we have to differentiate event sources. For all of these cases, processing a cloud of events is a choice. [5]

1.3 Viewing

In order to analyse a situation and have a view on what's happening, CEP offers a good approach that we are about to explain in this section.

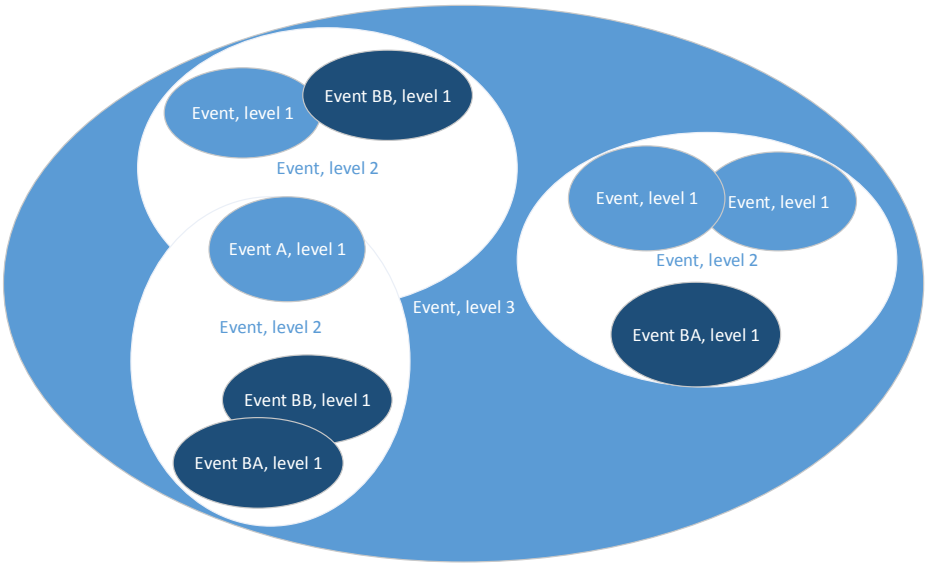
"To analyse a situation can be seen as analysing a complex event of higher level." [5]

1.3.1 Drill-down

A complex event is composed of lower-level events. Thanks to knowing the relationships between events, we can drill-down, meaning that we can zoom in and have a look at what the lower-level (complex) events represent. [6]

The other way around, we can also drill-up and look at events, which are being partly caused by a particular lower-level event. With this, CEP allows us to get a bigger picture. [6]

For a better visualisation, it's good if these relationships are portrayed graphically, so that one understands the meaning right away. In the picture 1 below, we see a complex event composed of lower-level events. Event A is a member of two level 2 events, there's no need for events to be disjunct.



Picture 1: Complex event composed of events of lower levels

1.3.2 Personalisation

One of the benefits of CEP is that it allows different event hierarchies to be applied and shown for different users. When looking at previous example again, a different person might create first event of level 2 from Events BA and second from Events BB. Then event of level 3 could be created by all of the Bx events, leaving all of the rest events irrelevant.

Every user or a group of them defines rules according to their needs and thanks to this, receives relevant information.

Human interaction with CEP system should provide a real-time view of data that can be altered using various attributes, parameters and values, and also analysed in order to obtain valuable and easily understandable information. One can clearly see that this is a first step on a DIKW hierarchy, from data to information (with knowledge and wisdom being next levels of this model). [10]

1.4 Various communication approaches

This section is built upon [3] and its very good description of various approaches for building and integration of any system. Interactions with and communication within system can be of more types:

- Time-driven
- Request-driven
- Event-driven

Time-driven interaction occurs at an exactly specified time, e.g. daily or hourly. The specification then says, if it is one-time or recurring event, et cetera. [3]

Request-driven interaction begins by a client (requestor), who requests a service from a server, provider, or similar. This communication is *synchronous*, client waits for the processing on server side to finish and for obtaining a reply. [3]

Event-driven interaction is open-ended and one-way. An event is created by an agent and here it ends. Other depending systems can react in any way, according to which rules or behaviour is going to be applied. [3]

As an example of sending a message (an email), if the interaction with agent is time-driven, the message would be sent every day at 10:20. If the interaction is request-driven, the message would be sent as a reply to our message (request). And thirdly, if the interaction is event-driven, the agent sends us a message every time a train passes the viaduct or every time it rains for more than 15 seconds. [3]

Interactions driven by either of two, time or request, could be described by an event. Event, defined by specific time on a clock or event of receiving an email. But since the mechanics of their processing differs quite significantly, it's good to differentiate these approaches completely, as time- or request-driven. [3]

2. Theoretical design

This chapter can be still considered as a theoretical part of the thesis, as it focuses on both theoretical and technical knowledge needed for a design of a CEP system. In more sections of this chapter, we describe options to implement some functionality. Therefore, we'll provide explanation of and differences between individual options.

2.1 Event Processing Agent

Event processing agents (EPAs) are objects in our graphs performing defined tasks or processing in-flowing events. They are the nodes of CEP architecture. Agents participate in one or more functions like [3]:

- Recognizing patterns of events
- Creating, reading or discarding events
- Aggregating lower-level events and matching them to higher-level events
- Detecting relationships between events with different creation time or source
- Performing calculations on event objects
- Triggering actions based on rules

2.2.1 Event objects

Format of event objects is implementation-specific. CEP suites written in Java, like Esper [31], use Plain Old Java Object (POJO) as carrier of information. POJO is a Java object not bound by any restriction other than Java Language Specification. [28] Esper and its properties will be explained in section 6.8.1.

2.2 Event-Driven Architecture

In this chapter, we focus on one particular style that is being used in regards to event processing. It is called *event-driven architecture*, often abbreviated as EDA. This way of defining architecture proposes using *minimally-coupled* event-driven components. [3]

Event processing software is here driven via receiving events in form of objects. Requirement for minimal coupling means that there's only a one-way connection between creator and processor of events. [3]

Implementation of EDA is suitable in case these features are present [3]:

- Discrete event occurrences are reported in real-time
- Event objects are *pushed* by their creator, not *pulled* by the processor
- Communication is one-way only and so the event creator does not wait for any kind of response
- Processor acts right after receiving the event
- An event is only a notification, it is not a specific command or any request

Mayor benefit of EDA is parallelisation, since events can be processed at different nodes, and no synchronisation is required.

Parallelisation of CEP is, on the other hand, not straightforward. There are, however, also some approaches. One is done by running different queries on different nodes or by splitting different operators of the same query among multiple nodes. MASSIF project combines these approaches on logical level. [4]

Second option for scaling of CEP is a Collaborative Fully Distributed Complex Event Processing (CFD-CEP), which builds a peer-to-peer mesh of EPAs and converts all event producers into CEP engines. Cooperation needed for actual processing is then achieved by selecting a coordinator per task. [7]

2.2.1 Comparison

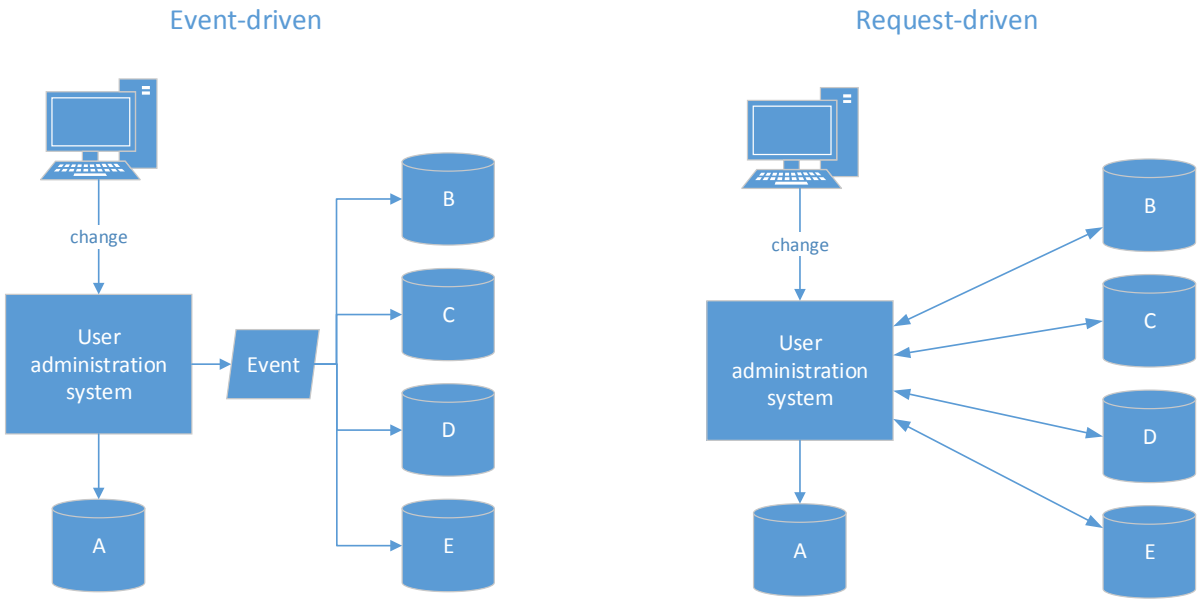
It is understandable, why continually working event-driven systems provide an outcome faster than the time-driven systems, which are run at a pre-specified time and process batches of transactions that have been accumulated. [3]

But the event-approach can have advantage over the request-driven systems also. Request-driven communication is dependent on server's reply. The server and network must be online and the communication in both directions must have a defined format. Any change with impact on this must be reflected in both/all agents relevant for given communication. With EDA, the communication is one-way only, the application sends notifications about events right as they happen and continue in its work without waiting for any reply. And as for changes, it's easier to implement them with minimal coupling, because of fewer dependencies. Thanks to this, EDA systems are agile and suit incremental changes better. [3]

Still, there are many cases, where there is a need to communicate synchronously, because we want the requestor to receive a response within particular conversation. An example can be credit card payment. If the funds are insufficient, the requestor needs to know this right in a reply received after checking. As a conclusion, in cases when a reply is **not** required, EDA approach provides benefits and is recommended. [3]

2.2.2 Data consistency

Especially in larger organisations with many database systems, EDA approach comes handy, as can also be seen in Picture 2. For example, if user changes his or her personal details in an administration system, in EDA architecture, new event describing this change is created and published in "fire-and-forget" way. That means, administration system lets others know about the change, but is not interested any further. This way, other systems record the change on their own. In request-driven architecture, administration system requests system or database B to record the change and after receiving acknowledgment, it continues and contacts C with same request. This continues until all data ale synchronised properly. [3]



Picture 2: Comparison of handling a situation in events or request-reply communication [3]

Important observations:

With event-driven architecture from this example, system does not know if there's a problem with recording the information about the change elsewhere. This responsibility should be delegated to an intermediary middleware agent that would handle delivery verifications, updates, retries if any system is not responding, etc. [3]

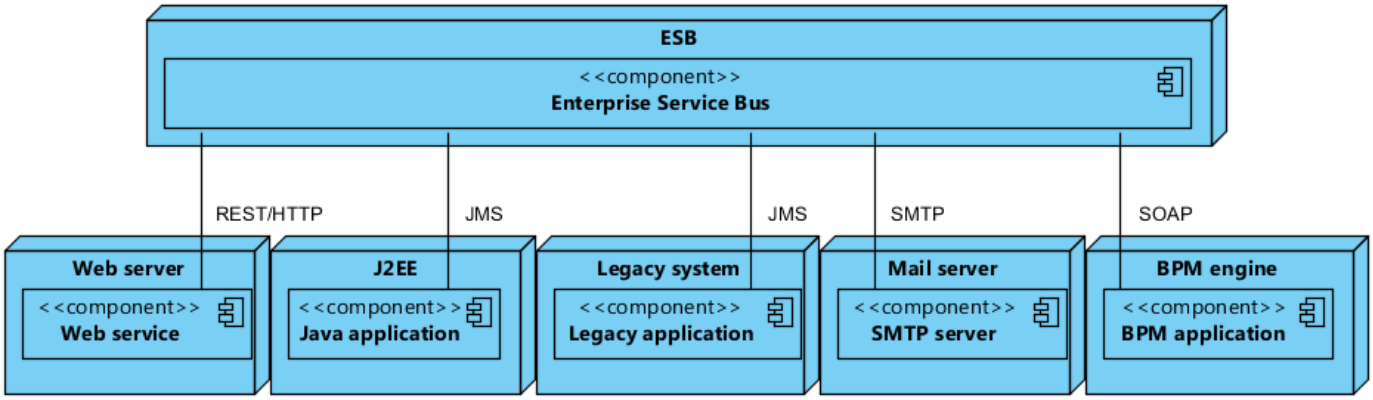
Generally, important benefit of synchronous systems is that the state is consistent and determinable, while asynchronous systems may possess a gap. A routing agent in middleware may solve this trouble to some extent and its responsibilities are listed in section 6.5.7.

One-to-many notifications are managed very well with EDA, thanks to its flexibility and high speed. Time-driven approach works for such systems, where there is no need to have data synchronised as soon as possible. Request-driven approach is suitable, if the number of receiving systems is small and these systems do not change often. The sender can then address recipient directly, and so the request-driven approach is needed for applications, where a piece of an information must be obtained in a server's response. [3]

2.2.3 Event-driven service oriented architecture

Service oriented architecture (SOA) is one of the widely-used approaches in software design, where every functionality is encapsulated in a service provided to system. Whole system is then created as a combination of various services communicating via enterprise service bus (ESB). [5]

Until these days, multiple approaches emerged and so the communication can be led asynchronously, both in request-reply model and in events. Combination of events with SOA leads to approach called Event-driven SOA, abbreviated as ED-SOA. Quite often, system architectures combine communication both via requests and events to exploit benefits of both approaches. [5]



Picture 3: Example of a service-oriented architecture showing services running on different nodes and communicating via enterprise service bus.

SOEDA is a unified development method proposed by [11] that helps to specify effects of complex events on process workflows. Method covers process specification, implementation and also execution. [11]

2.3 Operation

In this section, we'd like to outline basic features of operation of a CEP system. Since its inputs are events, it has to be *reactive*, meaning, that system decides how it is going to behave depending on the input events.

2.3.1 Patterns and rules

The decision-making should be *autonomous,* with as little user action as possible, and it has to be based on *rules* that specify reactions. They drive the process and are therefore called also *reactive rules*. Each rule has to have particularly stated trigger (event pattern we want to watch) and is able to generate events initiating next steps for the process (following actions that correspond to occurred trigger pattern). Typically, process is triggered if and only if events match specified pattern and their context is as expected.

2.3.2 Filtering

Optimal operation requires also filtering rules to be set. Filtering expects that there may be events that will be filtered out and not considered for further processing. Rules for filtering should be set and considered before heavy load of production data begins flowing in. Filtering may also impact results of CEP in both positive and negative way and should be therefore well-rethinked. [5]

2.3.3 Prioritisation

It is possible to assign a priority to every event and process them depending on that. Designer of such system should bear in mind the need to define rules in a way so that also events with low priority get processed and no congestion occurs.

2.3.4 Action

To achieve autonomy, we need to have defined steps for *exceptional* situations. The executions must be done in parallel and processes must be able to communicate asynchronously. [6]

After an analysis of data is finalised, we must be able to implement our decisions effortlessly. CEP system should provide on-the-fly process rules' modification, exceptional situations detection and handling, and full control over the process. [6]

Issues should be solved by a real-time action on the same level, where the given issue occurred. But finding the cause requires possibility to have a look at subprocesses or events that are lower in the *causal chain*. By a runtime drill-down diagnostics, we can find causal vector of the analysed event. [6]

Requirements of real-time analysis and on-the-fly system modifications allow us to exploit true purpose and strength of such system, because the user is always in charge and the system can adapt to current needs.

The action triggered by CEP rules can be of multiple forms. Rules can invoke:

- Simple black-box service
- BPM process
- Moving thread to different system

The black-box services, defined e.g. by WSDL, can be connected in order to perform calculations based on the defined input. Connecting BPM to CEP is then a way, how processes of higher complexity can be launched depending on complex events in current context. There are also intelligent BPM systems, which offer real-time monitoring of processes (BAM) or even events (CEP). [8]

2.3.5 Simulations and summary

One of the wanted features is also having a possibility to simulate CEP system's behaviour before application of new pattern or rule. This way, CEP analyst can define a rule and have a look at its outcomes before it is released to production.

Simulation and testing are crucial, as they help unveil errors created during analysis, design and implementation of rules. Since the rules are complex and assessing consequences is not often possible, simulations are highly needed.

To sum it up, CEP system must provide [6]:

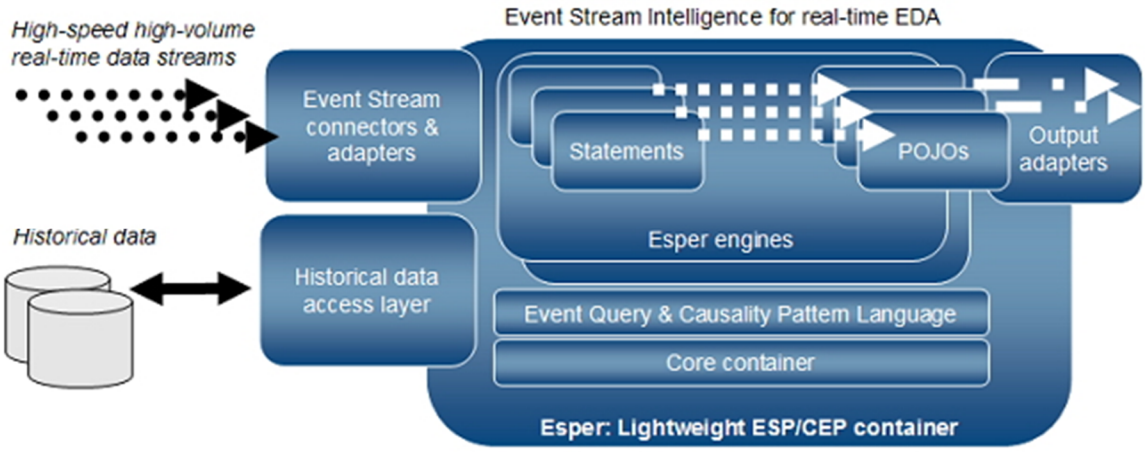
- Parallel and asynchronous communication between its processes
- Real-time viewing and data analysis

- Drill-down diagnostics thanks to knowledge of causality relationships
- On-the-fly rules' modification
- Exceptional situations handling
- Simulations

2.3.6 Real-time Business Intelligence

The key feature of CEP is operation in *(near) real time*, based on events. The actual data set available for processing is therefore changing with time *"under our hands"* and processing needs to reflect upon what is actual.

Picture 4 below explains processing of event stream. CEP is very similar, but with existing relationships between data (complex events). Also, multiple sources of data are usually connected to the processor creating network of higher complexity.



Picture 4: Explanation of operation of event processing by Esper. [31]

Apart from that, operation is similar and we can see main engine processing event objects (POJOs) based on statements. Processing is done upon in-flowing event objects and historical data available e.g. via OLAP (explained in section 5.8.2). Output adapters are connected to various services, depending on actual setup of triggers.

Thanks to this, and also, as is going to be explained further, business intelligence can be performed in real-time.

2.3.7 Operating in a sliding window

Since number of events is very high, technically, we are not able to record and keep all of them available for calculations. Running queries through vast amounts of data, and, at the same time, having correct answer in real time is not possible.

Therefore, a sliding window of reasonable length has to be chosen, a timeframe setting temporal bounds on events, we want to remember. Its actual length is result of a tradeoff between the quickness of response and availability of recent event history (and also costs and available technology). Load tests and performance assessment for various CEP software suites are therefore important.

Picture 5: Events within the sliding window are available for CEP.

2.4 Constraints

In order to have system in a compliant with all of the business rules, requirements and policies, there is a possibility to have constraints defined. They check for particular state conditions, values or behaviour. [6]

Tested conditions can be of two types [6]:

- Context – aim to test system state, temporal or causal conditions, etc.
- Content – tests of parameters of events

2.5 Predictions

Predictive monitoring is very useful as it can unveil problems and issues that are yet about to appear. [6]

If the system can analyse past events and problematic situations, find a pattern in the cause and search for this pattern in system's events, important predictions may be created. Any possibility of predictions is a great leap forward as this is a great opportunity for CEP systems in present and near future.

2.6 General architecture

Architecture of an Event processing network (EPN) consists of four types of components: event producers, channels, consumers and intermediaries. Producers, consumers and some intermediaries are event processing agents (EPAs). Flow of event processing is outlined here [3]:

1. Event processing begins with a producer detecting an event in its environment, creating describing object and emitting it in a notification message
2. Notification is then transmitted to one or more consumers via intermediaries

3. One or more consumers (also listeners or event handlers) receive event object and evaluate next steps depending on the processing rules and the input event. The decision may be to:
    a. Perform a task locally
    b. Invoke a service
    c. Trigger a business process
    d. Send a message
    e. Save the event for later processing
    f. Discard the event

2.6.1 Intermediaries

Intermediary is a go-through, transitional, component in EPN. Since an event can be transmitted inside a message, the intermediary can be a channel or one of two kinds of event processing agents (EPAs). Intermediary can be [3]:

- a simple channel transporting events, aware of message (i.e. packet header)
- router operating based on event's content, event-aware
- event-generating, as event was e.g. transformed or enriched with data

More on intermediaries or EPN can be found in Appendix.

2.7 Implementation characteristics

There are some features in which implementation slightly differs from ideal principles.

2.7.1 Immediate reporting

An event producer is supposed to act and sent the message just as the event occurs. In order to reduce network load, events are often packed into batches and sent together. Batches are of relatively small size (e.g. 10 events), so that the latency is kept under a second or two and requirement for quality of service is satisfied. [3]

2.7.2 Push or pull

Event producer always pushes events in direction to consumer. For the communication between event consumer and the EPN channel, there are more options to choose from. The channel may push notifications to consumer or they can be pulled from the channel. When a pull method is applied, it's important to add a timeout just in case the channel was not responding in order not to block consumer from processing. [3]

2.7.3 Sending messages

Event objects are sent through network in messages and the implementation can be done in multiple ways. Events can pass through a message-oriented middleware like ESB, via other applications or protocols similar to HTTP, SMTP or SOAP, or similar. [3]

2.7.4 One-way communication

In order to have implemented an event-driven architecture, we must held its principles. However, having only one-way communication in the system may often be limiting. An implementation can be done in a way that agents are coupled for some particular reason. These reasons may be that an event consumer, after receiving an event, contacts its producer because of a need to [3]:

- obtain more data from the event producer
- check correctness
- acknowledge receiving of an event
- perform a query on other producers

2.7.5 Immediate response

Event consumers in CEP systems do not always act immediately after receiving an event as it may be better to respond after more events are received and many calculations have already been performed upon the received events. [3]

2.8 Publish-and-Subscribe

Implementation of publish-and-subscribe suits the five principles of event-driven architecture mentioned before in this chapter. [3]

Name already suggests, how this method operates. An event is published and all of those, who are subscribed for it, receive it. Subscription is managed by a subscription manager, usually implemented into channel. Another option is that the subscriptions are managed directly by the event creator. Main benefit of having a well-specified subscription manager in the network is that both the creators and consumers of events may be added or removed at any time. Also, their subscription rules can be changed and anyone can have a look at dependencies between agents in the network and deduct valuable architectural information from that. [3] [29]

Broadcasting information about an event allows great flexibility as the consumer may be one, zero or there may be plenty of them. If the subscription rules are being checked in the channel, values in message headers are the differentiating ones. But if there is an intermediary EPA implemented to take care of this, it can have a look at an event in the message also. [3]

3. Applications of CEP

In this chapter, we describe applications of complex event processing. Firstly, we start with features demanded from a solution in real-world environment, then we describe usual domains, where application of CEP is usual. Last part of this chapter has form of a brief study on particular application of CEP in one of the business domains of CEP.

3.1 Benefits of CEP

Complex event processing does not always have to be done automatically and we all have experience with manual CEP, with data summarised by hand. Whether done automatically or not, processing itself can be described in 3 steps [3]:

1. Capturing events
2. Analysis
3. Triggering action – response

Automation of CEP brings more benefits, making this method appropriate for deployment [3]:

- Quality and precision
- Speed
- Filtering out unimportant
- Cost saving

The ability to solve data of higher granularity in short time is key in CEP.

3.2 System metrics

A lot of data is generated on daily basis and deriving conclusions and predictions from that brings useful advantage any enterprise or organisation can be beneficial of.

These six measures, abbreviated as REACTS, are important for design of any CEP system [3]:

- Relevance
- Effort
- Accuracy
- Completeness
- Timeliness
- Safety, security, privacy, reliability

3.2.1 Relevance

Processing rules in CEP must be designed in a way that only relevant information is shown to user. Otherwise, user may be easily paralysed by information overload. Also, end users of CEP solution should be able to control types of events, they are notified of and ideally get not more and not less notifications that allows them to operate

properly. Having only relevant information is the most important goal and the main struggle. Ability to deliver this is also the main metric, when decision about implementation of CEP system is about to be made. [3]

3.2.2 Effort

Implementing CEP solution requires lots of effort and the more user-tailored it is, the more effort is expected. Analysis of users' requirements is therefore integral part of development that should not be underestimated. Plenty of effort may also be spent after the final system is brought to production. Except for the maintenance, usability and learnability are important parameters to think of. [3]

3.2.3 Accuracy

Any EDA solution need to show accurate data in order to ensure proper functionality. That is not 100% possible and therefore there will be two types of errors known from statistics. [3]

For better explanation, let's have a look at these examples from network security:

Type 1 error (*false positive*): All of the network data suggest that the security was breached and an intruder got to a network location. But the intruder did not.

Our rule-based processor creates a complex event with critical warning and as an action, breach analysis and defensive measures have to be established. The hypothesis is positive (saying it did happen), but false (it actually didn't).

Type 2 error (*false negative*): All of the network data suggest that a breach did **not** happen, while it actually did.

Our rule-based processor **does not** create a complex event with critical warning and no people are aware of any security issue. The rule *missed*.

Similar examples of errors could occur e.g. in spam detection, credit card or identity misuse, monitoring of seismic activity and so on.

Errors are omnipresent, but for any real solution, the type of error presenting more danger needs to be minimised. In safety-related detections, error type 2 is usually more dangerous, but that is not a universal rule. It's good to have these errors in mind, when designing rules for a CEP system.

It is also important to set the tradeoff between accuracy and other metrics properly. Systems making practically no errors, need plenty of input, and therefore are usually not able to bring the information visible soon enough. Such alerts and predictions are of no value. [3]

3.2.4 Completeness

Every CEP system should be able to predict all of the important complex events that are about to happen. Design of the system and its rules should minimise probability

of false negatives. System's value would depreciate substantially in case it was not able to provide information about any important event. [3]

This is a point to think of, when rules are being designed and tested. Analysts and rule designers should not forget any important complex event and corner case that may occur and design the system as completely as possible.

3.2.5 Timeliness

In order to make a timely action, information predicted by a CEP system should be visible in reasonable time, depending on particular application. Usually, the more time is spent on data analysis, the less time to respond to an event remains, but, information produced is more accurate. Later in this thesis, we'll see timeliness playing diverse role for various systems. [3]

Since the actual data processing always takes some (nonzero) amount of time, we speak of having answer in *near real time*. That means with a very short delay, where the time was consumed by actual data processing. [18]

3.2.6 Safety, security, privacy and reliability

CEP is usually applied in domains of great importance. These systems often play important role in predictions, revenue generation, user and company data processing, or detection of breaches in safety and security of humans, property, etc. Building a secure and reliable system is therefore critical. Many event-driven applications collect plenty of user data in order to process it and offer a better service. In such cases, reason and special care during data handling and processing must apply. [3]

3.3 Suitability of CEP

Event processing is not always the best approach to build upon. There are ways to assess appropriateness of such implementation. Publication [3] describes a simple framework for assessing, whether one given domain is suitable for CEP-based solution.

Features of business applications, presence of which favours applying solution based on CEP [3]:

- Adaptability – presence of continual changes in environment, system or user's needs
- Instrument, monitor and record events – to foster improvements
- Responding to external events
- Exceptional conditions – ability to respond
- Unanticipated situations handling – ability to respond

If some of these features are required for the desired solution, we look at REACTS measures during the time of design and implementation. Of course, numbers like total expenditures and total cost of ownership must be thought over, just like for any other system deployment. [3]

It is also important to compare expectations of future system with what request-, event- and time-driven communication and processing can offer. If expectations are aligned with possibilities of given type, its implementation should be a good choice. [3]

For pioneering projects outside of usual CEP domains (to be mentioned in section 3.5), [2] also recommends to have well specified:

- Quality information model and description of possible states
- Time interval, until information necessary for analysis becomes available
- Rules to respond to actual situations
- Business value

3.4 Architectural patterns

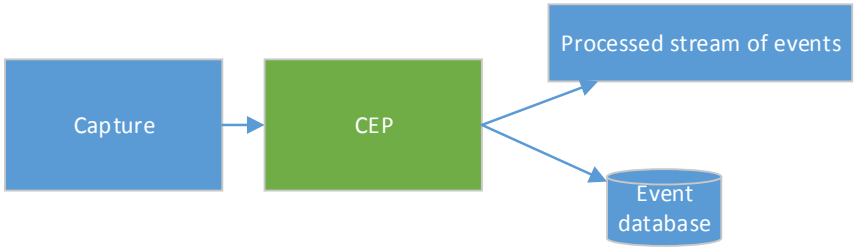
There are more ways or styles, how the CEP computation block can be put into the final architecture of the solution and what role are these machines going to play. The main five options are these [3]:

- Subsystem for event pre-processing only
- Monitoring system only (in Appendix)
- Monitoring of homogeneous systems (in Appendix)
- Monitoring of heterogeneous systems
- Solution built upon CEP

3.4.1 Event pre-processing

CEP machines, when used as pre-processors, listen to huge volumes of in-flowing event streams. At first, filtering and possible deduplication of events is applied. Events may then be reordered. At this stage, events can also be enriched with important data from other sources or simple calculations, like averages and totals, may be performed. [3]

Usually, these events are later stored in a database for later data analytics to be performed, or, forwarded for processing further, which will also be the case used in our solution.

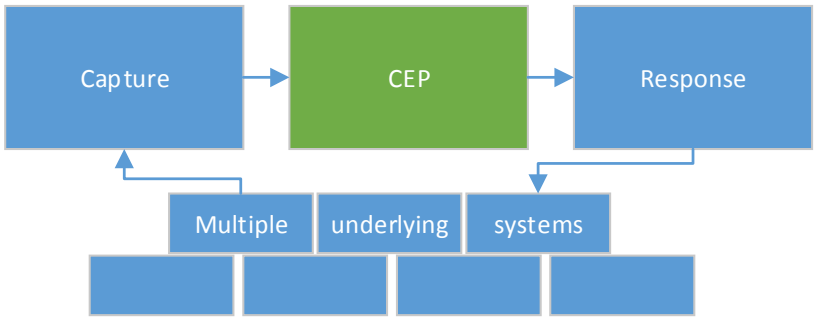


Picture 6: Event pre-processing [3]

3.4.2 Heterogeneous systems monitoring

With this approach, events are collected from multiple systems. The problem here is broadened, as we need to find connections between events from individual underlying systems.

In case that these are systems from various 3rd parties, translation into one format and getting stakeholders/partners to cooperate are tasks difficult to manage. [3]



Picture 7: Heterogeneous systems CEP monitoring [3]

3.4.3 CEP-based application

An application designed and created in this way is fully dependent on CEP. This pattern proposes application with rich functionality, where actions are decided by CEP and this idea offers an option for autonomous CEP system. [3]

Picture 8: CEP-base application [3]

3.5 Overview of CEP domains

There are multiple industries, where CEP is being applied. Size of markets and possible applications for CEP grow rapidly. Areas well-known for using CEP solutions today are [5]:

- Finance industry
- Security
- Transportation
- Energy
- Health care
- Consumer relations management, online sales and marketing
- Operational intelligence

- Military applications

In these industries, CEP is then used because of its ability to react quickly and to unveil knowledge that is hidden.

In applications, the complex event processing is used usually for [6]:

- Automation of (business) processes
- Marketplace transactions and their processing
- Systems to manage scheduling and control of any production line or synchronization of services
- Network monitoring
- Network and system intrusion detections
- Performance and throughput prediction

4. Technical Background

Purpose of this thesis is to describe application of complex event processing on a communication server used in ACME Company and possibly find new benefits and opportunities for deployment of CEP in given company.

Functions of the communication server, currently in use, include:

- providing updates
- operations with license numbers
- campaign management
- collection of statistics

4.1 Communication server terminology overview

This chapter contains basic terminology definitions. We tried to define main terms in context of communication server.

Communication server is a group of servers providing services to installations of ACME software.

License number (or also key) entered into application allows obtaining updates and certain services, depending on type of license number. It can be a free, trial or paid license. (Trial license offers user benefits of paid version for testing period, usually 30 days, and converts to free afterwards.)

Installation is an installed copy of ACME software on a single *device*. It may be an installation of free or trial version, where one license number is used by many installations. Or a purchased product with unique license number for given installation.

InstallationID precisely identifies one particular installation and serves as a key for communication between the installation and the communication server. An installation obtains its installation ID after first contact with the communication server and it can also change in cases like change of a product or license number.

As a *device*, we mean any particular piece of hardware, on which the installation runs. There is a need to have this tracked also as there may be more installations on one device. Typical examples are a personal computer or a phone with Android, Apple's iOS or Windows Phone operating system.

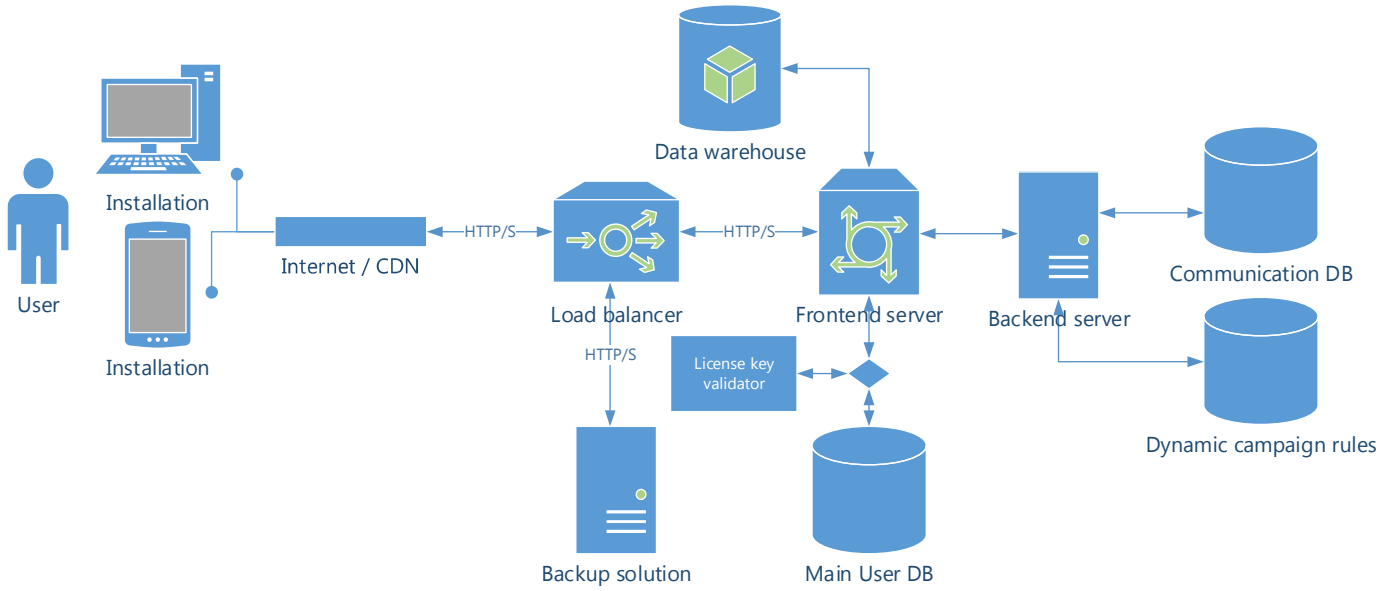
The *deviceID* is hence an identifier of the hardware upon which the installation runs.

4.2 Current architecture

For simplification, this thesis is written in a way, as if there was one communication server consisting of frontend server and backend server, which is designed to sustain load of all of the requests. The scheme can be found in Picture 9.

An installation contacts frontend of the communication server via load balancer that decides, which particular server is to be contacted. Load balancing is a widely-used method to ensure responsiveness and stability of any server online. Usually, there are more servers processing the requests or events (though acting as one) and architecture must be designed in a way to cope with synchronicity and data mirroring. And in case the frontend does not respond in required time, backup solution steps into the game. We shall not speak of the backup solution further, as it handles only critically important tasks.

Frontend server is connected to Business Intelligence's data warehouse, Backend server and the Main User DB, a database with users' data. Backend server is then connected to two important databases. Communication DB contains all data about installations, when records are made after 1st contact received from an installation. The other contains rules for campaigns that are to be shown on users' installations. More about campaigns and how they work will be explained in section 4.4 Marketing campaign.



Picture 9: Communication server

4.3 Communication and data

4.3.1 Request

Every installation connects the communication server regularly in order to check for updates. Installation sends a request containing data, mainly:

- InstallationID
- DeviceID
- Product and its version
- Build
- Operating system and its architecture
- License key
- Language
- Errors

If an installation contacts the communication server for the first time, we have event called 1st ping and a new record is created in database. With this, installation obtains its installationID, which is going to be used as an identifier in forthcoming communication. (This installationID may change e.g. with product upgrade or change of the license key)

Last state is stored in Communication DB, which adds checksum, timestamp, log, installationID, previously used installationIDs, and so on.

4.3.2 Response

In the response to installation's request, application may receive valuable info, to mention:

- New installationID (in cases like 1st ping, change of license number, …)
- Updates
- Campaign (if the dynamic rules decide so)

There are other values that can be transmitted in various attributes. We won't mention them as they are not crucial to understanding of communication server's operation and this topic.

Current architecture operates on synchronous communication over HTTP. That means that after a request from given installation is sent, it waits until receiving reply from communication server.

Later in this thesis, we'll have a look at possibility of this communication being driven in events.

4.4 Marketing campaign

By marketing campaign we mean a delivery of specific information to an audience, usually with intention to earn money or popularity by provisioning of marketed services or products to this audience.

ACME applications, provide functionality to display marketing or informational content in installations specified by campaign's target. The campaign is defined by *dynamic rules* that allow us to design behaviour of an installation to some extent and to show a banner, window or pop-up.

4.4.1 Dynamic rules

Campaign administrator sets rules for a campaign according to its specification. And that means, that every installation, which belongs to the target of given campaign, receives an instruction set with campaign definition. This definition triggers showing a pop-up or banner in application or displaying a window at a predefined time.

4.4.2 Example

Let's describe what happens, if we decide to target e.g. all of the users from country X with product Y and build Z. The dynamic rules in the belonging database are adjusted according to target's definition and some specific content (e.g. text, link, pictures) is added to message.

If user's installation pings the communication server, it receives response containing instructions for campaign and the message is shown accordingly.

5. CEP solution – business analysis

Practical part of this thesis is divided into two splits, analytical and design.

In this chapter, we are going to focus on analysis period of the whole project. We will therefore formulate basic requirements and business case for implementation of a system operating on complex events in ACME Company.

5.1 Business Analysis

Business analyst is a role in an IT project that is responsible for specification of *what* is to be done and *why* it is reasonable to have it done. The question of "what" is defined in scope, which contains a list of requirements, constraints and definitions of future solution. The "why" part should contain reasoning behind the decision to do the project or to fulfil particular requirement. Question *how* is answered to some extent only, as the lower level, implementation, details are left to discussions with system architects and an implementing team.

This chapter is heavily influenced by terminology and knowledge obtained from *Business Analysis Body of Knowledge [1]* as it is one of the cornerstone books for any business analyst.

Very often, role of business analyst serves as a discussion facilitator and translator between the business (project requestor), defining most of the requirements, and the implementing team, which provides expertise and capabilities and which will provide final realisation. [1]

According to [9], publication with importance similar to [1], business analyst does:

- Determine problem and identify business needs
- Manage requirements (this includes their discovery, documentation, prioritisation, …)
- Find and recommend viable solution
- Facilitate successful implementation of what was designed

In this thesis, we focus on determination of problem, identification of needs, requirements, and finding the solution. Project facilitation during implementation period will step in, after project receives a "go" decision and an idea turns into a running project.

5.2 Project: CEP implementation

Goal of this project is to implement a solution for processing of complex events happening primarily in the Communication server (as a mediator of communication with users' devices), but also from the eShop, mailing system, business intelligence, Main user database, service desk and customer support.

Events tracked from multiple sources must be identified and associated. Based on that, new business opportunities and knowledge should be obtained in (near) real-time.

5.2.1 Business need and Benefits

Collecting and processing data helps any company or organisation to know better itself and its business. With company-wide collection of data, new opportunities not known before, will emerge.

Having implemented event-driven architecture and exploiting benefits of complex event processing will allow company to assess current situation in near real time and act according to that.

5.2.2 Capability gap

Today, the data from eShop and the Communication server is recorded and tracked in order to derive meaningful information, but the data is collected in time-driven manner.

Also various types of data, collected company-wide, use different identifiers, and deriving conclusions from that is not automated in multiple cases. This technology offers ability to react right after obtaining a vital piece of information.

5.3 Project Stakeholders

Stakeholder is an entity that has an interest in given project. It can be a particular person, general role or a whole department. In such cases, it is good to have the whole department presented by particular person(s) that will act as a deputy in definition of the project.

5.3.1 Actors

For better understanding of stakeholders and their requirements, we've decided to use naming that is often used in agile development. Classes of users working with the solution will be described as *actors*. Agile development often creates also *personas* – humans, who seem as real persons with real names. Usually and archetype is chosen for better understanding of users' needs. [13] We give a name to each type of an actor, but not the persona details as this can be broadened later, when needed.

Business User

Beatrice is a business user and she is responsible for control of revenue in the organisation. She would like to see understandable graphs with the right amount of data showing trends related to money and user base. She needs to be notified of all of the complex events with significant impact as soon as possible in order to decide based on the assessment of complex events.

Project Manager

Lucas is a project manager and his responsibility is to deliver the whole project defined in scope. He is the main contact person for the business user. He plans the project and manages team dependencies with provided resources and time.

Business Analyst

Martin is a business analyst and, as was mentioned above, he should determine the problem and determine all of the needs and requirements, find and outline a solution and act as a technical guide during the process of implementation.

Campaign Specialist

Sarah is a campaign specialist and she sets and launches campaigns with advertising or informational purpose. Campaign's requirements are defined by the business user and designers. Possibilities are defined by current technical solution.

Business Intelligence Specialist

Michal is a specialist at Business Intelligence department and he wants all of the data being processed in real time with no interaction required. Similarly, all of the backups must be made automatically without any need of his interaction. He will play an important role during specification and implementation periods.

Data Analyst

David is a data analyst. His work is to understand collected data, assess their importance and create conclusions from them. He needs data with proper granularity in order to work effectively and to answer business's questions. He is able to plot graphs and show relations related to any data we have.

Administrators

Norbert is an administrator and he takes care of all of the databases related to backend of communication server. He also looks after physical servers in order to ensure stability, reliability and high performance of our systems.

Development

Adam is a developer and together with tester Beata, they work on the development of the communication server. They can add or remove functionality as well as define, how the system operates.

Beata's work is to check that specification is aligned with what is really required, everything is developed properly and the system runs according to its specification.

Christine is an architect and she designs systems in higher detail. The low-level architecture is created based on high-level design, discussions with analyst, the team and technical possibilities.

Legal and security

Wolfgang will act as a deputy of Legal department and his task is to ensure that all of actions made by any employee or system comply with law and also that no data or

knowledge we have gets to people, whom it should not. His interest is mostly in safety of user data, company's know-how, source codes and the financial data.

Customer support

Jimmy works as a support and he wants all of the users to be happy customers, who are confident with our products. He is interested particularly in knowing current state of things and hearing feedback from users.

Other stakeholders

The CEP solution will interact with another systems. And so besides the whole team mentioned, deputies of these systems will take some part in analysis, design, implementation, deployment and operation phases of this system.

Systems or departments, deputies (e.g. product owners), of which are going to cooperate on this project:

- Application development
- eShop
- Service desk

Event-processing rules Specialist

Operation of such new system is going to require a new analytical role dedicated to business rules management and to watch over smooth functioning of the whole new system.

5.4 Requirements analysis

A requirement is a condition or a capability that is needed in order to achieve an objective of given project. It must be *met* or *possessed* by the solution, so that the solution brings desired value. [1]

Usually, requirements gathering is an iterative process composed by requirements discovery, organisation, prioritisation and specification. In case the CEP system is about to be implemented, new iterations should be realised so that the scope is up-to-date.

5.4.1 Business requirement

The business requirements are high-level goals stated, describing objectives and needs standing behind this project. They describe the entire organisation, not just stakeholders within. [1]

Within this thesis, we've found these business requirements as the most value-adding and these functions must be present in future CEP solution:

Department	Requestor	Business requirement	Reason
Business	Beatrice	detect users bringing high revenue	monetisation
Business	Beatrice	offer adaptive system usable for user monetisation	monetisation
Business	Beatrice	provide real-time statuses in dashboard	monitoring

Table 1: Business requirements

5.4.2 Stakeholder requirement

Stakeholder requirements are expressions of needs of a particular stakeholder with overlooking dependencies or impacts on other stakeholders' needs and requests this might have. [1]

Except for the business requirements valid for the entire system, we are adding stakeholder requirements. We'll show and explain them in the following chapter.

5.4.3 Solution requirements

After requirements of business and various stakeholders are collected and known, they can be divided into two groups, *functional* and *non-functional requirements* of the solution.

For this project, we have also decided to use MoSCoW technique for assessing priority of requirements. Word MoSCoW is an abbreviation made of capital letters for must, should, could, won't (be implemented). We will therefore assess every requirement with a virtual grade depending on discussions with stakeholders, observations and information obtained from thesis's sources. Grades may help future readers to understand this analysis better. [15] [12]

Functional requirement

Functional requirements describe solution's capabilities and behaviour. They define services provided and a set of features, the solution is capable of. They can be divided as *user* and *system* requirements, where the user requirements define services provided to users and their operation with final solution. On the other hand, system requirements define functionality, but one that is kept in the inside of the solution and the user is just a consumer of the outputs of particular system functionality. [1] [12]

Functional requirements on our CEP solution are recorded in *Table 2*, marking also relevant requestor for further discussions about full specification.

Non-functional requirement

These requirements describe constraints and qualities for the solution to meet. They can be classified into three groups with focus on *product*, *organisation* and *external*. Requirements on product define qualities of final product, e.g. performance, usability and learnability. Organisational requirements define policies and procedures applied both during development and in real use, e.g. authentication will be done via password of length at least 10 characters or that the development will use Scrum

method. External requirements come out of the project's surrounding and these are most often legal and regulatory requirements. [1] [12]

Non-functional requirements on our CEP solution are recorded in *Table 3*, in the same way as for functional requirements.

5.4.4 Transition requirement

These requirements are needed only for the purpose of transition from current solution to the new one and will not be needed after the solution is launched and transition is over. [1]

We will not focus on transition requirements as it highly depends on implementation details. But generally, the projected Complex event processing system will be a new and added component into the whole net of information flow. Therefore, we expect that current systems will continue in their work and gradual changes in division of work, responsibilities, and communication schema, can be set.

Department	Requestor	Functional requirement	MoSCoW	Reason	Type
Business	Beatrice	detect users bringing high revenue	M	monetisation	system
Business	Beatrice	offer adaptive system usable for user monetisation	M	monetisation	system
Business	Beatrice	provide real-time statuses in dashboard	S	monitoring	system
Cust.supp.	Jimmy	know reason of user's contact/complaint	S	smooth operation	system
Cust.supp.	Jimmy	detect users making lots of complaints	M	user base preservation	system
Business	Beatrice	detect users, who bought particular set of products	M	monetisation	system
Business	Beatrice	detect users, who installed a product	M	monetisation	system
Business	Beatrice	detect users, who bought a product	M	monetisation	system
Business	Beatrice	act upon these detections	M	monetisation	system
Service desk	deputy	be notified of revenue (per time period) at level X	S	monitoring	system
Application dev.	deputy	user can shop in application	C	monetisation	user
Campaigns	Sarah	to know how the campaign is performing	S	smooth operation	user
Campaigns	Sarah	to build whole funnel	C	smooth operation	system
Campaigns	Sarah	ability to find a bug in a campaign	C	smooth operation	system
Campaigns	Sarah	overseeing campaign operation	C	smooth operation	user
Campaigns	Sarah	ability to contact user via email	S	communication	user
Campaigns	Sarah	ability to contact user via application	C	communication	user
Business	Beatrice	acquired apps - translation of original IDs to installationID	S	smooth operation	system
Business	Beatrice	users can view CEP operation, create and altered views	C	monitoring	user
Business	Beatrice	historical data available for CEP	S	smooth operation	system

Table 2: Functional solution requirements

Department	Requestor	Non-functional requirement	MoSCoW	Reason	Type
Cust.supp.	Jimmy	be promptly notified of user complaints (email, app, web, call)	S	user base preservation	product
Comm. server	Adam	seamless move of responsibilities between CS and CEP	S	smooth operation	organisation
Comm. server	Adam	provide updates via event-based communication	C	smooth operation	organisation
Application dev.	deputy	as little changes in application as possible	C	sustainability	external
Legal	Wolfgang	(internal) users have right level of information view available	S	security	organisation
Legal	Wolfgang	we comply with all regulations	M	security	external
InfoSec	Wolfgang	events are broadcasted inside company securely	S	security	organisation
InfoSec	Wolfgang	subscriptions for events are managed and approved at one place	M	operation	organisation
InfoSec	Wolfgang	backup and recovery scenarios prepared	M	security	organisation
Business	Beatrice	defined response time, uptime, downtime, maintainability	M	operation	product
Business	Beatrice	solution is scalable, reliable and of high performance	M	operation	product

Table 3: Non-functional solution requirements

5.5 Sources of complex events

In this chapter, we provide a list of all of the actors important for CEP system and for every actor, we mention important brief description, i.e. attributes, identifier used in communication and events originating at this place.

List of attributes is not exhaustive and for some sources, the number of attributes may differ. These attributes are not mentioned, since they were not particularly needed for our case. By not including some of the attributes to CEP, load of data can be lowered. This will be mentioned in section 6.3 on filtering.

5.5.1 Device

- deviceId
- Manufacturer
- Model
- Operating System
- OS architecture
- OS language
- IP address

Events: new device, OS updated, OS language changed

5.5.2 Installation

- installationId
- Product
- Product version
- Build
- Source
- Email
- Language
- License number
- Application bugs
- Errors encountered

Events: new installation, update requested, bug/problem reported, error occurred, campaign shown, campaign clicked

Application build for given installation can unveil also beta users or users acquired via company's distribution partners.

While the Communication server knows about particular device and installation, the Business Intelligence collects data about performance of desktop campaigns that are shown to users via their installations. Thanks to that, we have information on how many users were shown a campaign or what percentage of users clicked the offer.

5.5.3 Load balancer

- IP address
- Region
- Current traffic level
- Maximum traffic level

Events: under DoS attack, traffic reached level X, backup server used

5.5.4 Mail server

- email
- emailCampaignId

Events: email campaign sent, email sent, email opened, email clicked

As an email campaign, we mean a huge number of users receiving one email, e.g. with information or an offer, while the "email sent" event presents a single email sent to one user for reasons like an answer to user's query or transactional email sent after purchase or with a new password.

Events presenting opens and clicks on email happen in user's mailbox, but since the mailserver collects all of the responses, we can simplify, intermediary acts as a source in our system.

5.5.5 Communication server

- campaignId

Events: (desktop) campaign published - is available

Similarly to the Mail server, events happen in users' installations and the Communications server acts as an intermediary.

5.5.6 Web

- visitorId
- referralId
- visitorType
- duration

Events: new visit, build downloaded, banner clicked

Via Referral IDs, we are generally able to uncover information, from which source is the user coming. They are used widely in various systems including e.g. Google Play, store of Android applications. This functionality is also available in HTTP protocol under name HTTP referer and allows websites to track origin of user's visit for statistical purposes. [26]

Type of the visitor can provide us with info in case the visitor is a robot, e.g. indexing crawler or we don't want to count the visits.

5.5.7 Main user database

- orderId
- Name
- Surname
- Email
- Product
- Region
- Price
- Validity

Emails of users who register using their application are not stored directly in the Communication server's DB, but rather in this Main user database. Communication server's frontend needs to connect and manage this kind of data, when working with it.

5.5.8 eShop

- visitorId
- orderId
- Name
- Surname
- Email
- Product
- Region
- Price
- Validity
- Timestamp
- referralId

Events: product added to cart, product purchase, product purchase via campaign, validity of license prolonged

5.5.9 External application stores

- orderId
- Email
- Price
- referralID

Events: product installed, product purchase, product purchase via campaign

For external application stores, we mean services that provide packages (builds) for installation of ACME applications on mobile device. It can be e.g. Google Play, Apple's App Store, Amazon Appstore for Android or Appszoom.

5.5.10 Customer support

- orderId
- Email

- referralID

Events: new complaint obtained, refund request obtained, request for help obtained

5.5.11 Service desk

- ticketId
- requestId
- incidentId
- Creator
- Assignee
- Type
- Product
- Priority

Events: request created, request closed, incident created, incident closed, query sent

5.5.12 Event (in general)

- id
- source
- dateCreated
- tag
- user

Event: event (meaning actual occurrence)

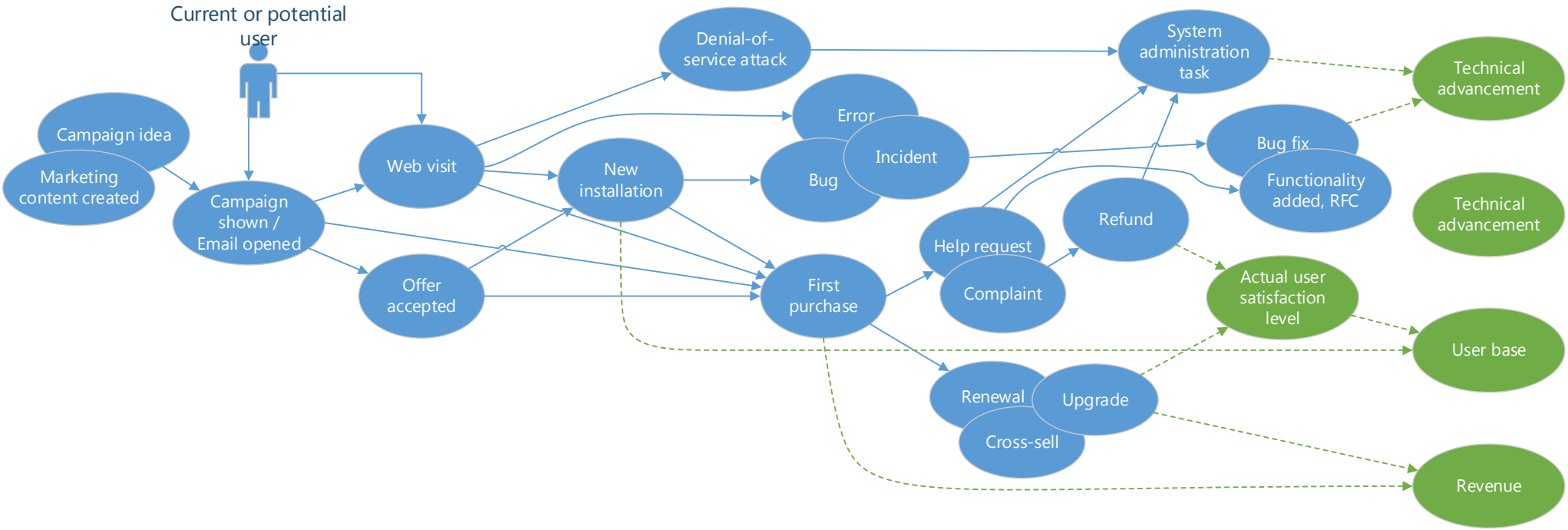
All of the events, no matter the source, have also these attributes mentioned above. They are general attributes we are going to need during processing.

5.6 Hierarchy of complex events

Knowing causal relations can be of great advantage. In this section, we'll draft an example, of how events can form a hierarchy.

Picture 10 shows actual relationships, where arrows mark relationships between complex events forming a directed acyclic graph. The piles of events, e.g. 3 distinct events one over another, are parallel events with same level in the hierarchy. They are put on a pile, so that the picture is condensed. Arrow leading to (or from) one of them leads also to (or from) all of them.

While events in Picture 10 are coloured in **blue**, we marked with **green** the business goals, e.g. to have big user base or low costs. These goals set company's directions, with dashed arrows showing main contributors. They can be converted to events, e.g. announcement of financial data or publication of report comparing quality of software with competition.



Picture 10: Complex event hierarchy as directed acyclic graph

Building a complex hierarchy is not an easy task to do, as there are too many connections and generalisations are in reality not straightforward, but the proposed scheme shows the most important complex events related to software distribution and achieving business goals. Practically every type from complex events visible in Picture 10 needs also trend and its evolution in time to be tracked. Huge change in trends is usually a new complex event, e.g. spike in revenue or drop of costs.

5.7 CEP suitability

In section 3.3, Suitability of CEP, we've mentioned few features that, if present, show, whether CEP is a suitable approach to design a system. Let's compare expected solution with these features to assess applicability of CEP.

5.7.1 Adaptation and exceptions

Continual change is present in multiple businesses, but, specifically domains that are sustaining rapid growth seem fit for CEP. ACME Company, as well as other business in the same domain, undergo many operational changes and also mergers with and acquisitions of other companies are rather present. With CEP, adaptation to such changes is easier to be expected and resolved. Users monitoring company's performance need to be offered a solution easily adaptable to change.

The processing rules that are yet to be defined must be exception-ready, as they exist in practically every real system and organisations need to adapt.

Right from the beginning, we should bear in mind a possibility, that there may be a situation not anticipated before, either because of overlooking it during analysis phase or because of an error in data and for similar reasons. CEP rules should be able to handle that in an appropriate way.

5.7.2 Externality and monitoring

Plenty of events in the designed system are going to be of external origin, e.g. an installation on a device, purchase of software in the eShop or visits on our web.

These events should be monitored and examined, processed (or filtered out) and outputs recorded. Records can help us broaden system's functionality and help in both, predictions and automated machine learning.

The solution we are willing to build, via this project, possesses features mentioned in section 3.3, Suitability of CEP. Therefore, CEP is a valid and suitable approach for the architectural design.

5.8 Technicalities

This section contains terms, methods and approaches that will be needed and used in the design chapter.

5.8.1 ETL processing

ETL abbreviation stands for *extract*, *transform* and *load* and refers to processes that usually run in systems, which collect data from various sources to perform data operations over them. Since the source data often uses different structure and identifiers, processing must be well thought over and the ETL processes must be able to communicate in multiple formats and with multiple databases. [32] [24]

That is also our case, though it ends with rules application and mining useful information from that. For the extract phase, connection and extraction of data sources must be designed according to functionality and possibilities offered by our solution. Transformation of data includes selection of important data, translation of values, so that only one classifier is used, joining and deduplication, etc. [24]

5.8.2 Viewing and OLAP

Online analytical processing, commonly shortened as OLAP, is a way to analyse multidimensional data interactively from more points of view, taking into account multiple dimensions. Thanks to that, data collected from various parts of the entire organisation can be pre-calculated and then handled and analysed at ease.

Since OLAP is widely used, CEP software suites offer option to connect to OLAP cubes and our CEP solution is therefore going to be connected to OLAP server of Business Intelligence, in order to obtain valuable statistics.

6. CEP solution – design

Second split of the practical part of this thesis will speak of design of a solution based on complex event processing. This chapter builds upon knowledge from all of the previous chapters, but mainly on requirements stated in previous, analytical, chapter.

"To *design* a system means to define architecture, its components and modules, interfaces and data of the designed system, so that it fulfils its requirements." [18]

6.1 Available sliding window

Actual number of days will depend on infrastructure, software solution deployed and filter set-up. Load tests should bring clarity into these questions.

For purposes of this thesis, we expect performance of our storage to able to handle all of the data created within 3 days from now, events are removed from memory the second their age equals 72 hours. Recent events are available and included into processing.

CEP software allow us to have a relational database with historical data available. And the Main user database suits this purpose well, as the ability to run queries on this database will allow us to include information we have about our users. Since this DB is too big, a narrow view on its data (email and related installations) suits the needs.

6.2 Patterns applied

Influenced by possibilities for CEP learnt from section 3.4, Architectural patterns, and bearing in mind, that our CEP solution for ACME Company needs to collect data from multiple systems that are not aligned and not using same identifiers, the best way is to design the CEP solution into more levels, which will:

1. Filter incoming complex events
2. Make connections between them via shared attributes
3. Trigger action depending on the rules
4. Store outcomes

From the mentioned patterns, our solution will resemble pre-processing (filtering level) and heterogeneous systems monitoring. The final solution will become, to some extent, a CEP-based application, as many communication server's responsibilities can be modelled as a reaction to complex events.

6.2.1 Rules notation

There are few notations that can be used for definition of CEP rules. Since event object represent an event that happened and created a new record, SQL-like languages are suitable for definition of processing rules.

Language used depends primarily on CEP software suite that will be applied. For purposes of this thesis, we've chosen *Event Programming Language* by iSpheres [23], because of its simplicity and understandability. Rules are defined this way:

on <trigger>
when <**Boolean** condition>
then <**sequence of** actions>

On is followed by an event that triggered the rule. Clause **when** specifies conditions that, when true, allow the rule to perform actions defined after **then** clause.

As a second option, created by Oracle, *Event Processing Language (EPL)* [19] is a mature rule definition language, which is SQL-like. Its syntax is as follows [16]:

```
[insert into other_stream_of_events]
 select select_list
 from stream_of_events
[where search_conditions]
[group by expression_defining_output's_grouping]
[having condition_for_searching_through_groupings]
[output constraint_for_throttling_of_output]
[order by expression_defining_output's_ordering]
[limit number_of_rows]
```

6.3 Filtering events

Not all of the events that happen are actually important for prediction and for evaluation of current situation. Therefore, message filtering can be applied in order to have better and faster working CEP solution.

At the level closest to event sources, we place pre-processors of complex events. Their main purpose is to:

- filter incoming event objects
- forward events for further processing in same format

One of the business cases, when translation to one format is needed, is when ACME Company acquires a different company using different IDs and logic in their data. The filter must translate application's attributes to company-standard format and create a translating table, where it will record new installationIDs assigned in a one-to-one correspondence to the former IDs of installations. The best way would be able to have this filter implemented directly in the Communication server, so that all of the systems on the backend side use only installationID as the identifier. In case this was not possible for any reason, this translation can be implemented into the CEP filters.

In filters, also attributes that are no longer needed need to be removed from event objects. By doing that, data load is lowered resulting in faster processing and accessing stored data.

on <event>
when <event.attribute **not in** attributesWhitelist>
then <**drop column** attribute>

Example above proposes whitelisting. Blacklisting some of the attributes would mean to drop all of those that are in a blacklist.

For such filtering functionality, filter has to be object-aware, as it must know attributes transmitted.

6.3.1 Filtering rules

For effective CEP, we need to filter out some of the unimportant events. Rules should be set to filter out:

- Incidents with lower priorities (with priority 1 being the highest)

  on <incident created>
  when <incident.priority >= 3>
  then <drop event>

- For incidents, we can also apply an exception. Since amount of them is relatively low, while their impact is high, these events should be available for CEP until they are resolved. CEP would then drop incident out of memory after catching relevant "incident closed" event.

- eShop purchases marked as 'testing'

  on <product purchase>
  when <event.source = 'shop' **and** event.order.type = 'testing'>
  then <drop event>

- Installations (or any other use) of ACME's software created by testers

  on <new installation>
  when <event.source = 'installation' **and** event.installation.type = 'testing'>
  then <drop event>

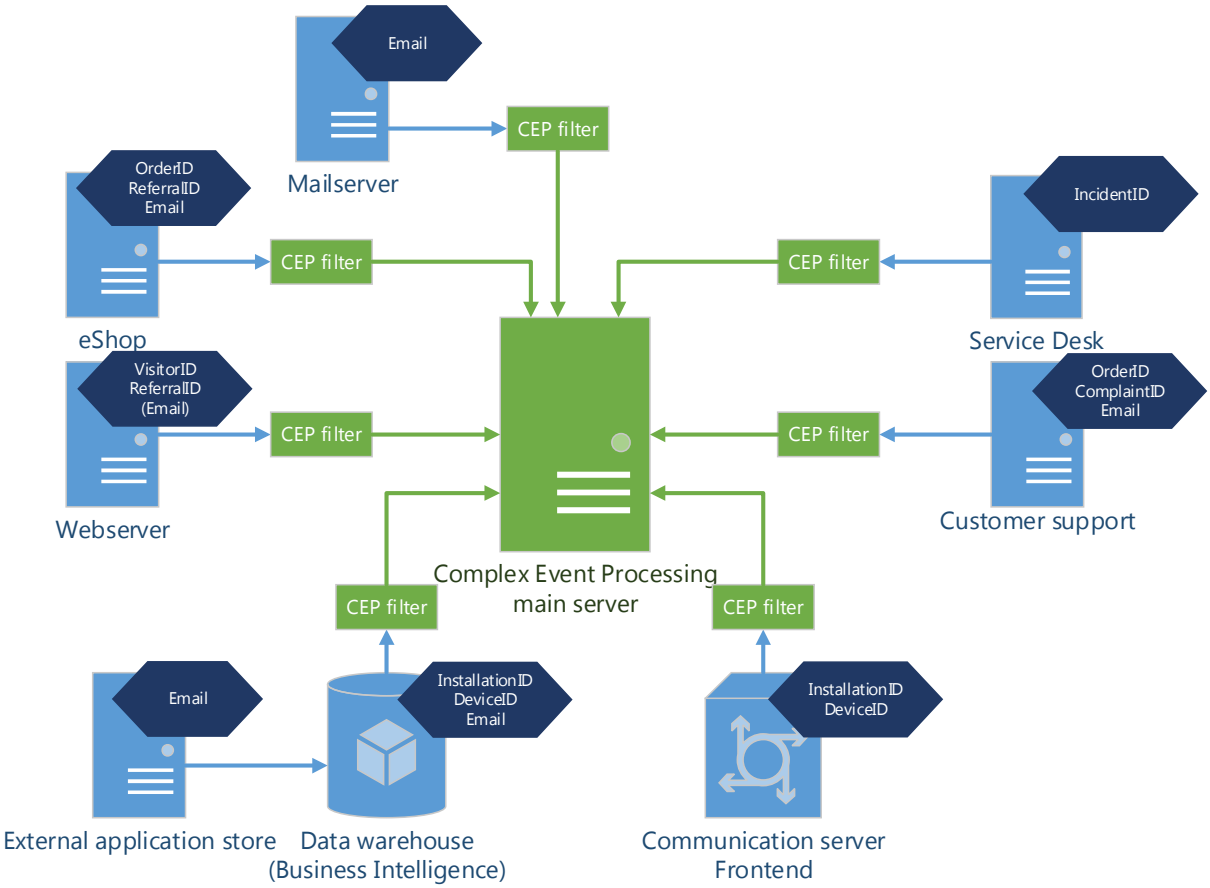
- Events and visits on web created by robots or users, who spent there less than 5 seconds

  on <new visit>
  when <event.source = 'web' **and** (visit.visitorType = 'crawler' **or** visit.duration<5)>
  then <drop event>

Filters have also power to enrich in-flowing events. All events coming from beta versions will obtain tag = 'beta' so that it is easier to build CEP rules aimed at beta users.

on <event>
when <event.source = installation **and** event.installation.build **in**

listOfBetaBuilds>
then <event.tag.add('beta')>



Picture 11: Filtering as a first step in processing input data

More use cases for filtering could be found during discussions and iterations. The need for filtering depends also on total performance of final solution.

Some solutions for CEP, like Esper [31], operate in a way that only specified types of events are collected, rest is filtered out. Whitelisting. In that case, rules should be rethinked from the other end to pick events of our interest.

The best would be to use software that allows both whitelisting (collect only specified) and blacklisting (collect all, except for specified). This way, one could choose an effective way for definition of rules depending on current event sources.

6.4 Creating connections, aligning events

After events reach common location in same format, we still have one obstacle to pass, in order to be able to join them. To be able to match events together, we need one common identifier.

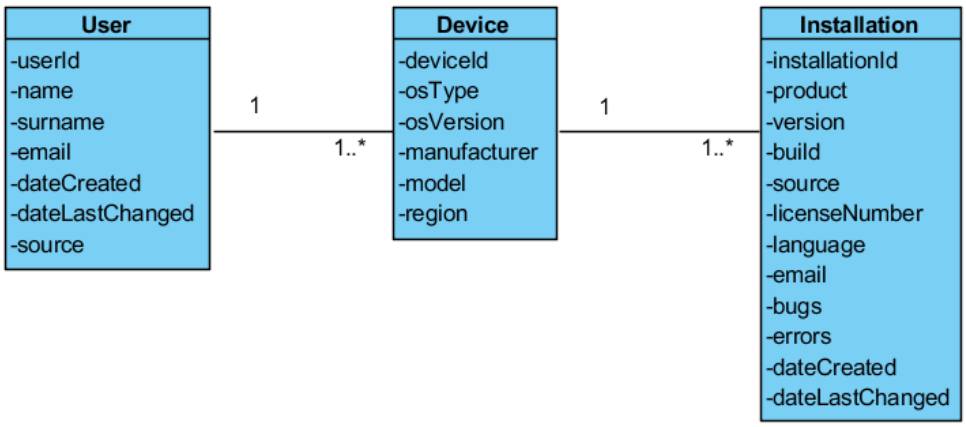
6.4.1 User, device, installation

Let's build on logic that a *user* can have one or more *devices* and every one of them can have one or more *installations* of our software. Relation between a device and an installation is always known from communication with the communication server.

But the relationship user-device is a harder question to answer. It is no problem, if a user has only one device. In that case, user and his or her behaviour is directly represented by events and communication from that particular device.

How to find all of the devices for any user? We are able to it, if on every one of user's devices, there is at least one purchased installation. User can be then identified by email.

If the user has a device with free/trial installations only, the way to identify him/her is to offer him/her a registration into an account. The account should contain information about all of the user's devices and therefore bind them together. Probably the easiest and also widely used method is to have email address serving as a key in case a data analyst need to match any user-related data.



Picture 12: Relationships between User, belonging Devices and their Installations

For the user with device without purchase or registration (without provided email address), data analyst's options are limited. Events of such origin will therefore provide less information and can be used for statistics.

Another possibility is that one person uses multiple email addresses. For example, if a user possesses two devices with two installations per each of one them, and also uses two emails for registration on both of these two devices, we could know that they belong to one person. Though current solution would have it hard to cope with that, it is possible.

But in case one email is used only with first device and the other email with second device only, there's no way to match these devices together and such case would be understood as having two distinct users. We consider this to be rather a corner case with low impact.

Partial solution to this is to offer user to log into account, where one can set email addresses used. Database beneath needs to record and bind together all of the emails used by one person. This is also useful in case, when user changes his or her email address.

6.4.2 Events from installations

Complex events occurring in any installation are going to be fed to CEP from the Communication server and the Data warehouse, managed by Business Intelligence. To link events from these two sources, we can use installationID, as this ID is used by both parties and both of them collect information from installations.

Linking these events with those from other sources is not a problem for users, who purchased and entered their license number. And for users, who registered but did not buy, linking events is possible via attribute email.

6.4.3 eShop events

Complex events with origin in ACME's eShop went through a purchasing process, where user enters his or her email address and are therefore easy to relate to other purchases. Every order is assigned its orderID for ensuring uniqueness.

If a user is logged into user account while shopping, companies can even send a reminder in case the purchase was not finished, so that the user can reconsider buying the product or service. ACME does not currently offer this functionality, as many users dislike having to log in, there are, however, possibilities how to make logging easier and more acceptable. This is going to be discussed in section 6.4.10, Single sign-on.

6.4.4 Customer support events

Customer support helps users in case of trouble or any unfamiliarity with software and its setup. Events of the highest importance and impact are complaints and requests for refunding a purchase.

If a user refunds a purchase with particular license number and orderID, it's easy to build a relationship with other events.

Complaints in ACME are reported via filling in a form on support's website, where email is requested and based on that, user is assigned a short and user-friendly code. This code is then used for the whole time of communication regarding complaint or any other help, no matter if the communication is done via telephone or any other means. Thanks to this, relationship can be built via email attribute also. Similarly to problem mentioned with devices and registered emails, in case the user uses different

email for filling of a complaint, it will remain standalone and can be counted to statistics only.

6.4.5 Short-term installations

With current technology, it is possible that a user installs application, for example into an Android device, opens it and uninstalls it within a short period of time, in which given installation did not have enough time to check for updates and was therefore not given a unique installationID.

In case we know nothing of this installation, there's nothing we can do.

There is, however, a possibility that source of the installation package would let us know of this installation. For installation of software via external application stores, like Google Play, Amazon Appstore for Android or Apple's App Store, user has to be logged-in in order to be able to start the installation. By processing the feed of emails from these sources to our Data warehouse, rule can found users, who installed an application, but never actually reached for updates.

In case of new users, who have no record in our data warehouse, this rule can do the counting. User is looked up via attribute email:

on <product installed>
when <(event.source **in** externalAppStoresList) **and** (event.user.email **is not null) and** event.dateCreated=now-24hours **and** user.device.installationId **is null**>
then <postInstallChurnNewUsers++>

Optionally, user can be asked for feedback via email provided in the external app store.

Next rule can be used for counting number of users, who have already been using our product and installed another one, but stopped using it before it even contacted the communication server:

for any (existingInstallation.product, newInstallation.product) **in** productList

on <product installed>
when <(event.source **in** externalAppStoresList) **and** event.user.email **is not null and** event.dateCreated=now-24hours **and** user.existingInstallation.installationId **is not** null **and** user.newInstallation.installationId **is** null>
then <postInstallChurnOldUsers++>

Since user's device has to be connected to the Internet during any installation of software from external app stores, waiting 24 hours might be even too long.

With settings in applications set, so that every installation would check for updates within few minutes after its first start-up, waiting time could be lowered.

On the other hand, user can install an application and not run it for quite some time. We think it is acceptable tradeoff to include into counts users, who may like the software, but did not have time to launch it. Thanks to this, waiting time for counting unsatisfied users can be lowered to tens of minutes, e.g. 60.

Having such counts in near real time would allow any company to understand much better its users' needs in rapid environment and mostly, whom the target audience consist of.

6.4.6 Service desk events

Service desk creates events, where incidents are of our concern. There is no need to build a relationship between an incident and particular user. But incidents on products or services must be taken into consideration during operation, so that the system adapts.

6.4.7 Mail server

Sending an email applies only to users, who provided an email address, e.g. during purchase or registration.

For sendout of emails in huge volumes, companies often use 3rd party service providers that take care of this mailings and actions related. For simplification, we will speak only of a mail server and won't get into details. Therefore, after the email template is prepared, we need to:

- Define and create target
- Send emails to all users in the target

Complex event, with mail server as its origin, can be an email campaign built up by events like sends, opens and user clicks on the email. Counts of these events per campaign are important metrics for assessing the performance.

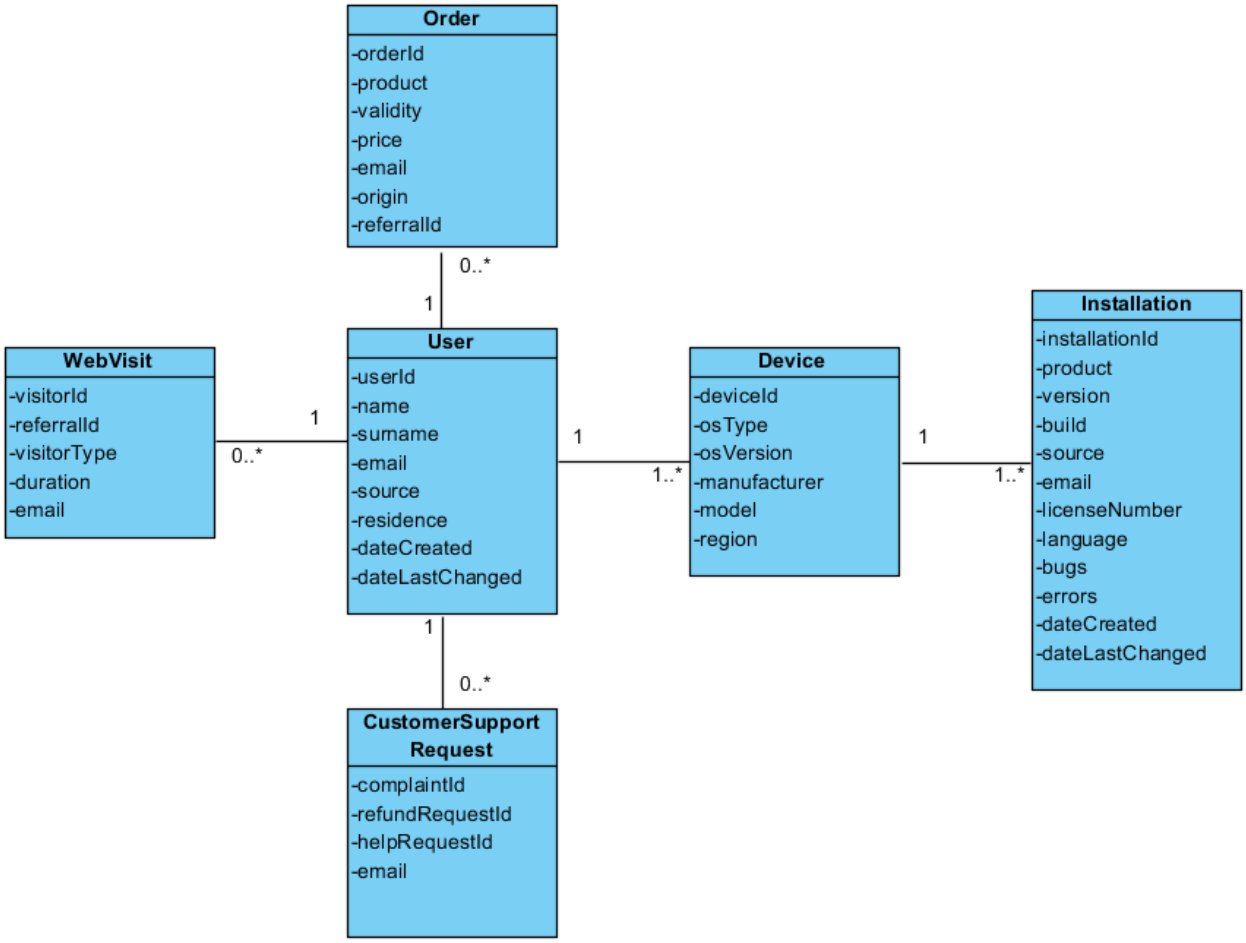
6.4.8 Website events

If a user is not connected into his or her belonging account and downloads a build for free installation directly from the web, we are not able to link this event to others. Events coming from anonymous browsing of web can be therefore good for statistics, like visits on a website and clicks on buttons, but that's all.

There are however exceptions, and more events could be tracked, if users visits ACME's website as a result of:

- clicking a button in an email
- clicking a button in any installation

This could be done by placing unique tracking parameters, referral IDs, behind buttons and their capturing on the ACME's web. The referral IDs represent email or an installation, which was the event source. We therefore have as much information about the event, as much we do about the email address or installation depending on where the user clicked. Of course, options are limited, but events like downloading a free build of product Y by a user of product X, can be tracked and knowledge of such events might be useful.



Picture 13: View on user-related data available in CEP

6.4.9 Website user flow

Next option, how to apply tracking of events on web can be done by recording "a whole journey," particular sequence of clicks on our websites. Aggregation of such data can show us, how majority of users behaves and can point out also user experience flaws on the web or in application. In case we know more about user thanks to redirection from an email or installation, or given user is logged into belonging account, these sequences can be recorded and analysed. Analysis per some particular user type could be later done with outcomes like, e.g.: paying customers like flow A, while customers of our free products like flow B more.

6.4.10 Single sign-on

Generally, users dislike having to sign on and then log in into every service they want to use, as it disturbs, from user experience point-of-view. An acceptable solution might be a practice called *Single sign-on*, when user logs into one system that ensures, next signing on or logging in will not be required. This is usually done by a single LDAP storage storing identities and authentication details and also not particularly possible in our case. [22]

Simpler and applicable approach, but one that is not a single sign-on according to its definition, is to use a service provided by Facebook, Twitter, LinkedIn or OpenID. With e.g. Facebook Connect or OpenID on our web, user has option to log in with Facebook's password, but the need to register is omitted. Since user base of these social sites is vast, this often requires only a single click of user. [27]

Requiring user to log in with OpenID or identity from a social site would allow us to know user's identity and to build upon that knowledge, in case it is connected with other data we have. Impact on experience and usability, of our services would have to be tested. Final decision should be made depending on that and include recommendations, where to apply this form of authentication.

6.5 Architecture

With input data filtering explained in section 6.3, we shall now present the whole architecture for the complex event processing system in Picture 14.

6.5.1 Entities

In the table 4 on the next page, we are providing description of entities, the graph is composed of.

As for entities in the graph, systems present in ACME Company are marked with **blue** colour. **Grey** colour marks entities considered external. The rest is the CEP solution, with **green** used for its building blocks and **pink** to mark the CEP decision rule engine that triggers reactions.

Entity	Description
User	Customer of ACME Company, client whom we deliver our software and services
Installation	One particular copy of a product installed on one device
Business user	Any person monitoring CEP operation and outcomes in ACME Company
Workstation	Entry point for a business user to get to CEP dashboard
Dashboard	Overview of actual counts, statistics, metrics, depending on the user-defined view
eShop	Particular website, on which users can browse and purchase products and services offered by ACME Company
Webserver	Website of ACME Company publicly available on the Internet
Service desk	Company's department ensuring friction-less operation and handling issues
Customer support	Single point of contact for users of ACME Company regarding any praise, complaint or similar
Business Intelligence	Department responsible for operation of data warehouse and collection of respective data
Complex Event Processing	Projected solution for analysis of current status and future trends
CEP DB	Supportive database, storing important CEP outcomes
CEP triggers	Module in Complex Event Processing that triggers action based on reactive CEP rules
Mail server	Machine for handling emails and campaign mailings
CRM	Machine handling preparation of targets for campaign communication via email or applications
Email DB	Supportive database for Mail server
Communication DB	Supportive database for Communication server Backend and CRM
the Internet Content delivery network	Public information network CDN makes network communication faster thanks to optimised packet forwarding
Load balancer	Network component ensuring optimal distribution of load onto more servers placed behind
Backup solution	Separate server handling requests in case the communication server is overloaded
Communication server: Frontend	Server handling communication with installations, external servers and databases
Communication server: Backend	Server ensuring communication with data-layer, databases and CRM system
Dynamic campaign rules	Simple database containing definitions for communication between installation and communication server
Main user DB	Database containing data of ACME's users
License key validator	Service for assessment of correctness of license numbers

Table 4: Entities of CEP solution

6.5.2 Architecture relationships

Operation of current solution, in bottom part of graph, was already explained in chapter 4. To remind, installation contacts the communication server by accessing its frontend in a request for updates, that is further directed to backend server,

processed, and reply is sent to the installation via frontend again. Last state of user is stored in the Communication database and in case the request-originating installation is suitable for a campaign, backend server adds instructions for the campaign to show up depending on how it is defined in Dynamic campaign rules database.

CRM system is connected to databases of both communication streams, email and desktop, and creates a layer above this data. From this view, targets for campaigns are later defined.

User can also browse ACME's web, where there is a possibility to manage one's email subscription. That means, user can ask for emails to be sent to him or her, or opt-out from communication.

User can then visit eShop, which is a separate entity available on the web, and purchase products or services offered by our company. User can also create a complaint, request a refund or help with our products and services.

Events are then fed into CEP machine as was described in section 6.3 about filtering, where they are connected and processed.

Real-time outcomes are fed to the CEP Dashboard available to business users of CEP system. Statistics are also sent to Business Intelligence for possibility of further analysis and processing.

CEP machine has its belonging database available for saving processed events, their hierarchies and outcomes for later use.

Based on the rules that are yet to be described, CEP can trigger action in various systems and services and if the rules decide so, this solution is able to:

- adjust content of the web (add advertising banner and call to action leading to eShop)
- adjust prices and offers (cross-sells, up-sells)
- trigger alerts in the dashboard, service desk and customer support in case of mayor issue
- create incident at service desk, which relates to alerts mentioned above
- send an email or notification directly to a particular user
- adjust campaign rules in order to target bigger user base

Data available for CEP:

- User-related data (installations, eShop, external app stores, customer support, web)
- Historical user data (view on Main user DB)
- Statistics (Data warehouse)
- Infrastructure (load balancer, network, backup)
- Service desk (incidents)

6.5.3 Interfaces

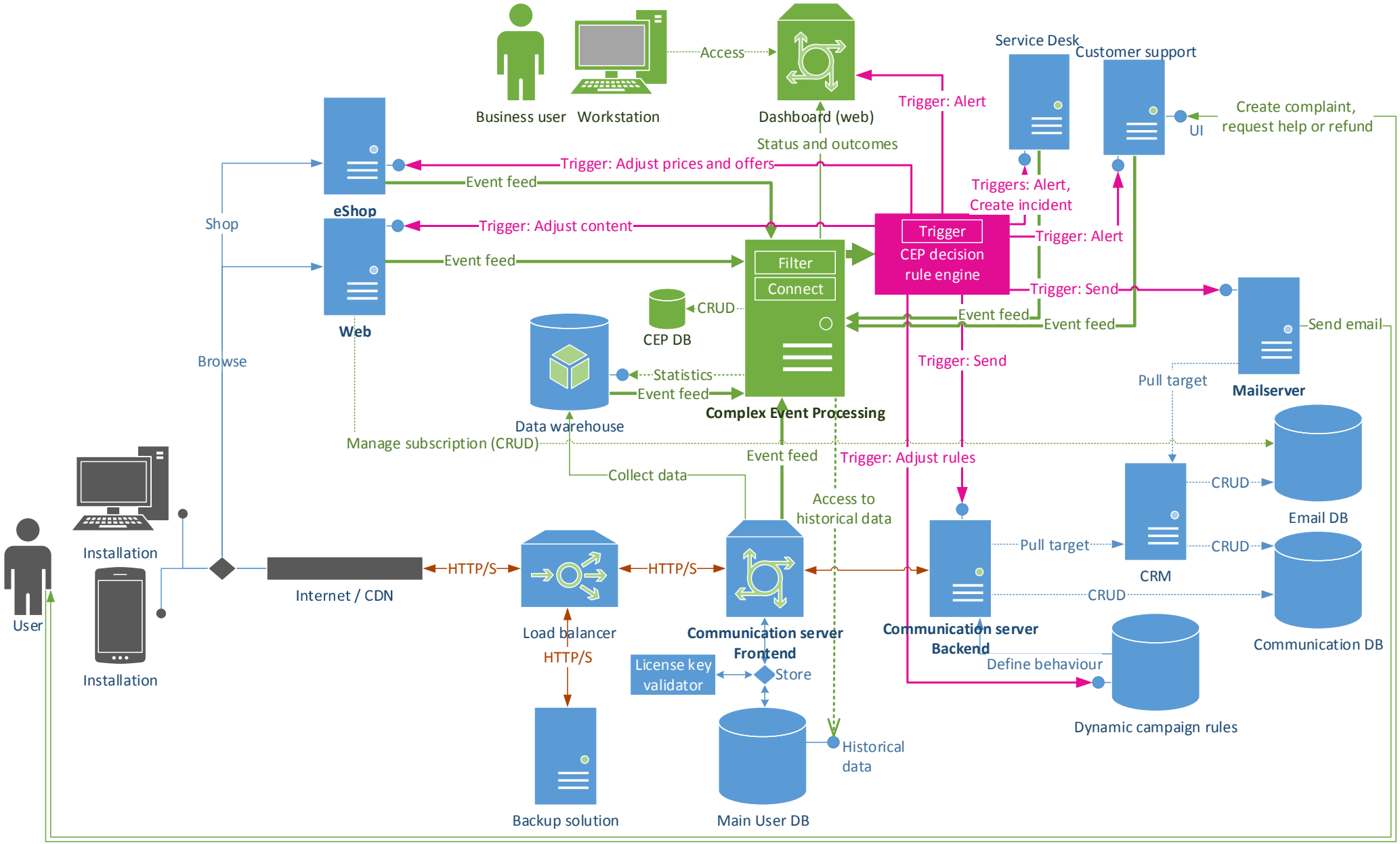
Most of the interfaces are understandable right from the architecture shown in the next Picture 14. Currently used systems need to provide an API, CEP system can use to reach data located in these systems and to use services provided by them.

In case CEP triggers an action to alter e.g. content in eShop, request containing XML definition must be sent.

There are interfaces to BI's OLAP cubes provided, so that complex information is available company-wide.

The Main user database needs to provide a view on historical user data to CEP and therefore provide an interface allowing quick searches.

Mentioned are various types for an API, while the CEP suite used needs to be able to communicate with all of these common interfaces.



Picture 14: Architecture for processing of complex events

6.5.4 Request-reply vs. event-driven communication

Request-reply way of communication between system's components is still going to be used in the proposed architecture, as it suits better if response *is required*. This type of communication is marked with **blue** arrows.

Communication between user's browser and eShop (or webserver) will be driven by requests from user, over HTTPS. But as the order is created, event containing information about order is created and moved for further processing.

6.5.5 Event-driven communication

Communication via events is preferred for one-way communication and is marked with **green** or **pink** arrows in the graph, where the **pink** colour is used for events triggered by CEP solution as a decision, usually result of processing.

The communication between the Communication server and the installation itself needs to cover provision of InstallationIDs, updates and campaigns. We've marked it with **orange** as this channel could be used for communication in both ways, events and requests.

While it is best to provide new InstallationID directly in a reply, both updates and campaigns can be distributed in events. With update or campaign published, respective event is published available and all of the installations would be subscribed and would download needed data from an accessible location.

6.5.6 Subscription manager

Since the delivery of event objects is based on publish-subscribe method, there must be a place to set the subscription rules. The Subscription manager assigns to every source of events in our system, list of all of the subscribers that should be listening.

These rules can be saved in belonging event sources, but the subscription manager is a central register and the only place where the changes can be made. In case these rules are altered, they are transcribed into the event source.

6.5.7 Routing middleware agent

In order to have a reliable communication within system, we need a middleware agent that will:

- verify successful delivery
- retry in case a subscriber is not receiving relevant events
- route event objects

This functionality needs to be supported by the CEP software suite and embedded in our EPN.

6.6 Triggering rules

The core idea of complex event processing is to *collect* all possible information (in event objects) and to *act* upon knowledge of everything that is happening in the *sliding window* available. In this section, we focus on rules that define and trigger action.

Requirements stated in previous chapters apply to most of these rules. For a good design of rules, it is good to have REACTS properties on mind. That means relevance, effort, accuracy, completeness, timeliness, safety and security.

6.6.1 Detect users bringing high value

Any user can bring ACME value in more ways, while not all of them are expressed directly by money. Value for ACME can be brought in these ways:

- Purchasing a product
- Testing beta versions
- Providing feedback
- Spreading good word

Spreading good word about ACME outside of its realm is something, we could hardly include, but, if a user comments on ACME's forum via account with his/her email address, it is possible.

We created next example to show, how the perceived value added by our user could be calculated:

Event	Calculation	Value (€)
user buys software X for 20USD	= price	20
user buys software Y for 40USD	= price	40
user uses beta version of product Z	= 5	5
user reports two bugs in software X	= 2 * 11	22
user reports three bugs in software Z	= 3 * 5	15
user provides feedback on usability of Z	= 1 * 10	10
user shares an idea for new feature in software X	= 1 * 35	35
Total valueAdded		147

While information about purchases, beta users, ideas or feedback can be made available for the available history, less important events become unavailable as the time passes.

Running this query could be performed in a time-driven manner, while it outputs a list of users bringing value higher than threshold set:

select user **where** valueAdded(user) >= threshold

For a better reactiveness, the rules above can be triggered by any event considered to be value-adding. As an example, next rule sends a present to valuable customer and ensure, it is not sent twice:

> **on** <product installed>
> **when** <valueAdded(user) >= threshold **and** user **not in** giftedUsersList>
> **then** <sendGift(user) **and** giftedUsersList.add(user)>

Limitation here is the length of the sliding window and therefore, the rules need to be combined with search in historical data.

6.6.2 Detecting new installations

New installations connect to Communication server in order to get newest updates. Though we may have no information (like email) about user yet, if the backend does not find record about this installation, new installation ID is assigned. Events "product installed" need to be monitored in a dashboard as an important metric.

> **on** <product installed>
> **then** <sendWelcomeMessaging>

6.6.3 Adaptive monetisation

The power of CEP is in instantaneous ability to act and since every company is dependent on in-flowing money, ability to adapt can mean a huge benefit. But what are the possibilities to adapt in regard to pricing and monetisation overall?

Let's assume that the number of purchases per hour is more than 10% below or above (exponential) moving average for that hour in comparison with our sliding window.

(Moving average, from statistics, is a calculation, where we take last *n* values for a variable and calculate their average. For exponential moving average (EMA), values' weights are in exponential distribution, last value has the highest weight.)

CEP system can watch the amount of purchases and trigger a command to lower or rise currently offered price by 1€ for a limited amount of time.

> **if** <eshop.purchase.countInTimeInterval(now-1hour,now) < 0.9 * EMA(eshop.purchase.countInTimeInterval(now.hour-1,now.hour))>
> **then for all** <product **in** productList **do**(product.price = product.price – 1;)>

Function EMA calculates average from amounts of purchases within selected time interval of day and does this for all days in the available sliding window (3 days).

This rule could be broadened in very same way to offer more adaptability, e.g.:

- 10% above (price = price+1)

- 17% above (price = price+2)
- 22% above (price = price+3)
- 17% below (price = price-2)
- 22% below (price = price-3)

Such rules definitely need to have defined exemptions as one would not want to have these rules applied e.g. during Christmas time. Also it is good to check feedback and opinions of users as well as legal advice before applying adaptive pricing. Users might not like the idea resulting in lower sales or these changes might not be allowed to be performed as desired in some jurisdictions.

Company offering transportation services called Uber had constraints not defined properly, when prices rose four times during times of danger. This resulted in public outrage and damaging Uber's reputation. [21]

6.6.4 Adjusting web content

Content shown on web should be updated automatically, if rules decide so. Here are few points, where such updates would be helpful:

- Prices shown on various websites, after their automatic change in eShop
- Better selling products could be placed higher and be more visible on the web
- Users logged in should not be offered purchase of products they already own
- In case of known product bug with high-impact, the product could be temporarily placed less visible, until problem is solved
- Frequently Asked Questions on the web of customer support could alter automatically according to which questions were the most asked

For such adjustments, all of the depending services have to offer an option to be altered. If they do, adjusting instructions could be simple. For order of products shown, they can have a score that says, how visible should the product be.

on <incident created>
when <incident.type = product **and** priority = 1>
then < **if** <web.product = incident.product>
then <web.product.score = 0>>

6.6.5 Send email or application notification to a set of users

Thanks to connection between CEP (triggers) and both Mail server and Dynamic campaign rules, we are able to send an email to a specific set of users. To define the target for a sendout of emails, we need a list of email addresses, usually in simple CSV format. More data like name, surname and license number are available and needed in cases, when we want to greet a user personally or inform him about ending validity of his license.

Notification in user's application can show up. After CEP rules decide that particular installationId, or a set of them, should receive some messaging, new rule is created in Dynamic campaign rules DB. Next time user contacts the communication server, installation receives specified message.

6.7 Other requirements

Final technical parameters depend greatly on the chosen CEP software and implementation. Generally, the solution should possess:

- High speed processing of high-volume complex events
- Quick delivery of results
- Serviceability
- Safety and security

6.7.1 Backup and recovery scenarios

CEP solution and its data should be mirrored and backed up in order to be ready for unpredictable occasions.

In case this system must have an outage for any reason, events created within defined sliding window can be stored in a parallel location. Then, they would get into CEP after system is initialised again.

As system is designed to analyse current situation and optimise user monetisation, dynamic campaign rules must be set ready for operation with CEP system out of order. After CEP solution is launched again, it can regain control over the distribution of campaigns.

6.7.2 Real-time dashboard

CEP software suite should offer clear and dashboard filled with important metrics in near real-time. Such dashboard should show trends in:

- User count
- Revenue
- Installs, uninstalls and purchases
- Complaints, bugs
- Incidents
- Website and eShop traffic
- Campaigns (as an aggregation of various metrics)

Dashboard should offer functionality to build personalised views and to drill-down to have a look on particular campaign, revenue per application and region, etc.

Today, there are multiple producers of software for data analytics and easier dashboards that, with right data, are able to visualise values and trends. Products of

Datazen [30], GoodData [33] or Tableau [34] could be used in case dashboard of our software solution was not meeting the needs.

6.8 CEP Software

There are existing pieces of software designed for complex event processing that can be used for our implementation.

6.8.1 Esper

Esper is an open-source solution created by EsperTech Inc. [31] that offer CEP capabilities. Esper is written in Java and can be embedded into other standalone applications. It uses Event Processing Language and is able to run continuous queries, joins, filtering and computations. Esper can be integrated with Java, .Net, Map and XML events. Triggers are then sent as POJOs. [31]

Historical data in standard relational database can be joined via JDBC, so that running queries on history is possible. [17]

Thanks to multithreading, Esper is able to process 500 000 event/s with 1000 statements (rules) in the system on one computer with standard processing power (2GHz dual core CPU). [17]

Enterprise Edition offers a secure and fault-tolerant platform with more features, including [31]:

- Clustering and scalability
- High-availability and resilience (named EsperHA)
- JDBC server endpoint, enables connected clients to run queries
- Push messaging REST service built on JMS
- Remote management via JMX connector
- Configurable web-based graphical user interface

Ability to scale Esper is highly needed and performance tests could show, whether it fits our needs. Clustering and high-availability features are signs that Esper is suitable for our needs.

6.8.2 Apache Camel

The Apache Software Foundation stands behind routing and mediation engine suitable for enterprise integration. Apache Camel uses EPL from Oracle as its language. On its webpage, Apache offers two ways of work with Camel, and to [14]:

- Integrate Camel with Esper
- Use Camel RX (based on reactive extensions)

Reactive extensions provide an API that operates on an asynchronous stream of events. Apache Camel is therefore another option to be considered. [14]

6.8.3 TIBCO BusinessEvents

TIBCO Software Inc. creates multiple products with aim especially on company and data integration, analytics, cloud and event processing. There are multiple publications that describe setup of their CEP suites, solution architects can find inspiration in [2].

TIBCO BusinessEvents offers a full software suite for CEP and so is good to be considered. [20]

7. Conclusion

This diploma thesis on analysis and design of deployment of Complex event processing (CEP) begins with description of theory and application of CEP.

Then, the current solution in ACME Company is described, so that the thesis can build upon current systems and data.

After first obstacle, connecting the data, was overcome, we propose both, rules searching for information of business value, and, example actions that can be triggered as a result of rules' evaluation. As a next step, in addition to rules from section 6.6, more rules need to be defined and set, so that information hidden inside complex events is revealed.

Along with that, outlined architecture needs to be elaborated into low-level detail of formats, in which data is stored, communication protocols used, system dependencies, et cetera.

Deployment of CEP in ACME Company or any other similar organisation has potential that should not be neglected. Multiple application possibilities of this powerful method can be found in any system, and since CEP can unveil facts about oneself, as well as one's environment, its importance will continue to grow.

Bibliography

[1] ANALYSIS, International Institute of Business. *A guide to the Business analysis body of knowledge (BABOK guide)*. Version 2.0. Toronto: International Institute of Business Analysis, 2009. ISBN 9780981129228.

[2] BROWN, Paul C. *Architecting complex event processing solutions with TIBCO*. New York: Pearson Education, 2013, xxiv, 278 p. ISBN 978-0-321-80198-2.

[3] CHANDY, K and W SCHULTE. *Event processing: designing IT systems for agile companies.* New York: McGraw Hill, 2010, xvii, 235 p. ISBN 978-0-07-163350-5.

[4] GULISANO, Vincenzo and Ricardo Jiménez PERIS, Marta PATIÑO-MARÍNEZ, Claudio SORIENTE. *Event processing engine architecture*. MASSIF [online]. 2011 [cit. 2015-05-21]. Available from: http://www.massif-project.eu/sites/default/files/deliverables/D3.1.1%20-%20Event%20processing%20engine%20architecture.pdf

[5] LUCKHAM, David C. *Event processing for business: organizing the real-time enterprise.* Hoboken, NJ: Wiley, 2012, xiii, 273 p. ISBN 978-0-470-53485-4.

[6] LUCKHAM, David C. *The power of events: an introduction to complex event processing in distributed enterprise systems.* Boston: Addison-Wesley, 2002, xix, 376 p. ISBN 0-201-72789-7.

[7] NGUYEN, Filip and Tomáš PITNER. *Scaling CEP to Infinity*. In *Proceedings of 9th Summer School of Applied Informatics*. 2012. ISBN 978-80-210-6058-6.

[8] PALMER, Nathaniel. *iBPMS: Intelligent BPM systems: Impact and Opportunity.* Lighthouse Point, FL: Future Strategies Inc., 2013, 220 p. ISBN 978-0984976461.

[9] Project Management Institute, Inc. *Business analysis for practitioners: a practice guide.* Atlanta: Project Management Institute, Inc., 2015. ISBN 978-1-62825-069-5.

[10] RANCE, Stuart. *ITIL service transition.* London: TSO, 2011, xii, 347 p. ISBN 978-0-11-331306-8.

[11] WIELAND, Matthias and Daniel MARTIN, Oliver KOPP, Frank LEYMANN. *SOEDA: A Methodology for Specification and Implementation of Applications on a Service-Oriented Event-Driven Architecture.* Poznan: Springer, 2009, vii, ISBN 978-3-540-79395-3.

[12] BÜHNOVÁ, Barbora. *PB007: Software Engineering I*. Brno, 2014. Information: https://is.muni.cz/predmet/fi/podzim2014/PB007?lang=en

[13] Personas: An Agile Introduction. *Agile Modeling* [online]. 2014 [cit. 2015-05-22]. Available from: http://www.agilemodeling.com/artifacts/personas.htm

[14] Camel CEP. *The Apache Software Foundation* [online]. 2015 [cit. 2015-05-22]. Available from: https://camel.apache.org/cep.html

[15] MoSCoW Prioritisation. *DSDM Consortium* [online]. 2015 [cit. 2015-05-22]. Available from: http://dsdm.org/content/10-moscow-prioritisation

[16] EPL Reference: Clauses. *EsperTech* [online]. [cit. 2015-05-21]. Available from: http://esper.codehaus.org/nesper-4.1.0/doc/reference/en/html/epl_clauses.html

[17] Technical requirements. *EsperTech* [online]. 2015 [cit. 2015-05-22]. Available from: http://www.espertech.com/products/technicalspec.php

[18] Federal Standard 1037C, Glossary of Telecommunication Terms. *Institute for Telecommunication Sciences* [online]. 1996 [cit. 2015-05-22]. Available from: http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm

[19] Overview of the Event Processing Language (EPL). *Oracle* [online]. 2014 [cit. 2015-05-22]. Available from: http://docs.oracle.com/cd/E13157_01/wlevs/docs30/epl_guide/overview.html

[20] TIBCO BusinessEvents. *TIBCO Software* [online]. 2015 [cit. 2015-05-22]. Available from: http://www.tibco.com/products/event-processing/complex-event-processing/businessevents

[21] This Is How Uber's 'Surge Pricing' Works. *Time* [online]. 2014 [cit. 2015-05-22]. Available from: http://time.com/3633469/uber-surge-pricing/

[22] SSO Benefits. *University of Guelph* [online]. 2015 [cit. 2015-05-22]. Available from: https://www.uoguelph.ca/ccs/security/internet/single-sign-sso/benefits

[23] Event Programming Language. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/Event_Programming_Language

[24] Extract, transform, load. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/Extract,_transform,_load

[25] event. *Wiktionary: the free dictionary* [online]. San Francisco (CA): Wikimedia Foundation, [cit. 2015-05-21]. Available from: https://en.wiktionary.org/wiki/event

[26] HTTP referer. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/HTTP_referer

[27] Single sign-on. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/Single_sign-on

[28] Plain Old Java Object. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/Plain_Old_Java_Object

[29] Publish-subscribe pattern. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2015-05-21]. Available from: https://en.wikipedia.org/wiki/Publish-subscribe_pattern

[30] *Datazen Software Inc.* [online]. 2015 [cit. 2015-05-22]. Available from: www.datazen.com/

[31] *EsperTech Inc.* [online]. 2015 [cit. 2015-05-22]. Available from: http://www.espertech.com/

[32] *Business Intelligence - Data warehousing - ETL* [online]. 2015 [cit. 2015-05-22]. Available from: http://www.espertech.com/

[33] *GoodData Corporation* [online]. 2015 [cit. 2015-05-22]. Available from: www.gooddata.com/

[34] *Tableau Software* [online]. 2015 [cit. 2015-05-22]. Available from: http://www.tableau.com/

List of abbreviations

API	application programming interface
BAM	Business activity monitoring
BPM	Business process management
CDN	content delivery network
CEP	complex event processing
CFD-CEP	Collaborative Fully Distributed Complex Event Processing
CRM	customer relationship management
CRUD	create, read, update, delete
CS	Communication server
CSV	comma-separated values
DAG	directed acyclic graph
DB	database
DIKW	data-information-knowledge-wisdom
EDA	event-driven architecture
ED-SOA	event-service service oriented architecture
EMA	exponential moving average
EPA	event processing agent
EPL	Event Processing Language
EPN	event processing network
ESB	enterprise service bus
ETL	extract, transform, load
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	identifier
ITIL	Information Technology Infrastructure Library
JDBC	Java Database Connectivity

JMS	Java Message Service
JMX	Java Management Extensions
MASSIF	MAnagement of Security information and events in Service Infrastructures
MOM	message oriented middleware
LDAP	Lightweight Directory Access Protocol
ODBC	Open Database Connectivity
OLAP	online analytical processing
POJO	Plain Old Java Object
REACTS	reliability, effort, accuracy, completeness, timeliness, security
REST	Representational State Transfer
SMTP	Simple Mail Transfer Protocol
SOA	service oriented architecture
SOAP	Simple Object Access protocol
SQL	Structure Query Language
UI	user interface
WSDL	Web Service Description Language
XML	Extensible Markup Language