

Dealing with Data Streams: Complex Event Processing vs. Data Stream Mining

Moritz Lange[1], Arne Koschel[1], and Irina Astrova[2(✉)]

[1] Faculty IV, Department of Computer Science, University of Applied Sciences and Arts Hannover, Ricklinger Stadtweg 120, 30459 Hannover, Germany
moritz-lange@outlook.de, akoschel@acm.org

[2] Department of Software Science, School of IT, Tallinn University of Technology, Akadeemia tee 21, 12618 Tallinn, Estonia
irina@cs.ioc.ee

Abstract. Recently, data generation rates are getting higher than ever before. Plenty of different sources like smartphones, social networking services and the Internet of Things (IoT) are continuously producing massive amounts of data. Due to limited resources, it is no longer feasible to persistently store all that data which leads to massive data streams. In order to meet the requirements of modern businesses, techniques have been developed to deal with these massive data streams. These include complex event processing (CEP) and data stream mining, which are covered in this article. Along with the development of these techniques, many terms and semantic overloads have occurred, making it difficult to clearly distinguish techniques for processing massive data streams. In this article, CEP and data stream mining are distinguished and compared to clarify terms and semantic overloads.

Keywords: Data streams · Complex event processing · Data stream mining · Data mining

1 Introduction

Nowadays, massive amount of data is generated by plenty of different sources. Smartphones, social networking services and the Internet of Things (IoT) are just a few examples of these data-generating sources. This trend is often referred to the term *Big Data*. Data is valuable. In some cases, even a company's value is measured by the amount of data it keeps and collects. Therefore, processing and analyzing data is a key challenge of modern businesses [7,16].

Over time, data generation rates have increased rapidly. Nowadays there are almost continuous flows of information - data streams. Due to limited resources, it is no longer feasible to persistently store all the incoming data. Computer systems, and in particular data analysis techniques, must adapt to these new conditions to meet the needs of today's businesses. For example, streams of transactions need to be analyzed in real time to be able to respond to credit card frauds [1,10,16].

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12252, pp. 3–14, 2020.
https://doi.org/10.1007/978-3-030-58811-3_1

In the last 20 years various techniques have been developed to deal with massive data streams [10]. These include complex event processing (CEP) and data stream mining, which are examined in this article. Along with the development of these techniques, many terms and semantic overloads have occurred, making it difficult to clearly distinguish techniques for processing massive data streams. In this article, CEP and data stream mining are distinguished and compared to clarify terms and semantic overloads.

The remainder of this article is organized as follows: Sect. 2 focuses on the basics of data streams. Section 3 shows the central concepts and characteristics of CEP. Furthermore, Sect. 4 introduces the general techniques of data mining and shows how they can be applied to data streams. This is commonly referred to as data stream mining. In Sect. 5, both techniques CEP and data stream mining are compared. Section 6 summarizes the results and gives a recommendation as to when each technique should be used.

2 Data Streams

As mentioned in the introduction, CEP and data stream mining are both techniques for dealing with massive data streams. Since data streams are the central concept of both approaches, this section defines the characteristics of data streams and shows the differences to traditional data.

According to Henzinger et al. [15] a data stream is defined as a sequence of data items $x_1 \dots x_i \dots x_n$ such that the items are read once in increasing order of the indices i. In combination with the work of Gama et al. [13], the following characteristics of a data stream can be defined:

1. The data items in the stream arrive online.
2. The system has no control over the order in which the data items arrive.
3. Data streams are potentially unbounded in size.
4. Once an item from a data stream has been processed, it is discarded or archived.

The first fundamental difference to traditional data is that there is no random access. Data items arrive in an online fashion. A second characteristic is that the system has no control over the order or timing of their arrival. Especially with several distributed data stream sources it is not possible to predict when which source will send how many data items. As the definition of Henzinger et al. has shown, another difference to traditional data is that there is a potentially infinite sequence of data items and not a finite set. Since it is not feasible to store data items of the stream persistently, there is a one-pass constraint. Once a data item has been read and processed, no further (fast) access is possible. This is because memory and computation time are restricted when processing data streams. The one-pass constraint is the key challenge that needs to be overcome by an algorithm to guarantee real time processing [1,3,13].

Due to the potentially infinite length of data streams, besides the four characteristic differences to traditional data, there are also potential differences in the content of data:

Concept Drift
Concepts in data streams may evolve over time. Unlike traditional data, there is no finite number of static concepts. Statistical properties and relations between data items can change during the runtime. An illustrative example showing a concept drift is the analysis of e-mail traffic. Especially the detection of spam mails shows that concepts in data streams are not always static. For example, if the author of a spam mail notices that his mails do not pass through the mail filter anymore, he will change some words in his mail to trick the filter. Therefore, the concept has drifted. Because concept drifts can occur anywhere in data streams, handling them is a key challenge for many algorithms and systems [1,12].

Massive Domain
Another aspect is that the potentially infinite size of the data stream implies a potentially very large domain. Especially in data streams, discrete attribute values can have a large number of distinct values. This phenomenon can also be seen well in the example of the analysis of e-mail traffic. For example, if the goal is to determine communication routes, a record must be kept for every pair of e-mail addresses. Considering the amount of possible e-mail addresses, the task is not trivial. Such discrete attribute values with huge ranges are very common in streams, because data items usually have individual identifiers [1,12].

In summary, data streams differ in many ways from traditional data. In particular, algorithms computing on data streams must be able to handle the special characteristics. Stream processors like Apache Spark (Streams) or Apache Flink are implementations that deal with exactly these requirements, which is why CEP and data stream mining systems often rely on such a stream processor.

3 Complex Event Processing

After introducing the basics of data streams in the previous section, this section will introduce a technique for processing massive data streams, evolved from active databases in the 1990's and early 2000's - CEP [9].

3.1 Events

The central concept of CEP, as the name suggests, is the Event. According to David Luckham [18] an event is an object that is a record of an activity in a system that has three characteristic aspects:

1. **Form:** The form of an event is an object, which may have attributes or data components. For example, if a system writes to a logfile, it could generate a log event containing the content of the log message as an attribute.

2. **Significance:** An event signifies an activity. The form of an event usually contains data describing the activity it signifies. In the above example, the log event signifies the activity of the system log. Events can signify any activity, which is why events in literature are also defined as records of anything that happens.
3. **Relativity:** An event can be related to other events by time (an event may have prior to before another event), causality (an event may have been occurred in consequence to another event) or aggregation (an event can consist of several other events). The form of an event usually encodes its relativities.

A complex event is an aggregation of other events. Complex events become visible with CEP. They are usually not explicit. A complex event could be, e.g., a series of suspicious transactions which are individually uncritical but together constitute a credit card fraud.

3.2 Event Streams

As mentioned above, CEP is a technique to deal with massive data streams. In the context of CEP, data streams are considered as event streams. From the definition of a data stream (see Sect. 2) it is easy to derive the definition of an event stream by replacing the concept data item with the concept event defined above. The goal of CEP is to understand and interpret the relativity of events in event streams. In other words, to find and respond to complex events near real-time [9].

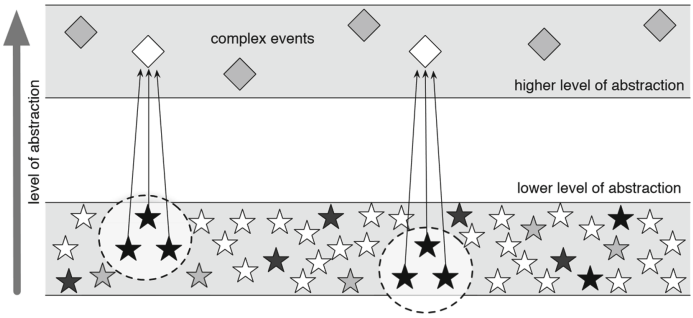

Fig. 1. Event streams and complex events [5]

Figure 1 shows the relation of event streams and complex events (streams). CEP systems analyze the stream of events of a lower abstraction level to find events on a higher abstraction level. CEP systems follow the sense-process-respond cycle (see [4]), which means that they respond to complex events. This response can be a new event representing the complex event or any activity (e.g., a service call).

3.3 Processing Event Streams

Having introduced the central concepts and objectives of CEP in the sections above, this section discusses how implementations of CEP systems can process massive event streams.

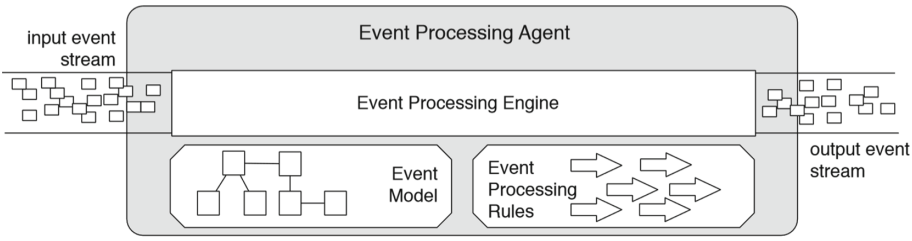

Fig. 2. Architecture of CEP systems [5]

Figure 2 shows the basic architecture of CEP systems. It can be seen that CEP systems belong to the rule-based systems. Rules can be defined declarative in event processing languages (EPL). Rules that are placed in the CEP system are always structured as follows: ***CONDITION*** $P(e_1e_2 \dots e_n) \rightarrow$ ***ACTION*** $action(e_1e_2 \dots e_n)$. If the event pattern P occurs, the CEP engine will execute the action. All events that may occur are known a priori by the event model.

As mentioned in Sect. 2 the length of data streams is potentially infinite and the resources are limited. This requires a method to allow the data stream to fit into the memory for processing. A sliding window is used for this purpose. The idea is to restrict the considered events to the last recently occurred ones. In general, a distinction is made between a length window, which takes into account the last N events, and a time window, which considers the last events in a fixed time interval [5,9,18].

In summary, CEP is about matching predefined structures in data streams. The concept of the event abstracts from the raw data. Rules are defined in an EPL and executed by the CEP engine on a sliding window. When a rule fires, it will result in a new event or activity. Typical applications for CEP include Business Process Management or Algorithmic Trading. In these cases, the patterns are known a priori. However, CEP (alone) does not cover the case if the patterns are unknown.

4 Data Stream Mining

After CEP was introduced in the previous section as an established rule-based approach with a priori known patterns, this section presents data stream mining as a technique for extracting previously unknown patterns and knowledge from data streams. Compared to CEP, data stream mining is a relatively new technique that has only become useful for practical applications by the advances in hardware and software over the past decade [10,12].

4.1 Mining Data

As the term data stream mining suggests, there is a connection to data mining. Therefore, the following is an introduction to the basics of data mining.

According to Han et al. [14], data mining is the process of discovering interesting patterns and knowledge from large amounts of data. This process can be applied to many different data sources. Traditionally, data mining is operated on data sources with random access. These include, for example, databases or data warehouses. There are four major tasks that occur repeatedly in data mining applications, so that the following four problems are considered as the fundamental ones in data mining [1][14]:

Association Pattern Mining
The task of the association pattern mining is to identify (data) items that imply the appearance of other items within a transaction. If a certain item often occurs together with others, association rules can be deduced. An illustrative example of the use of association pattern mining is the mining of transactions in an online shop. The transaction data can be used to determine which products were often bought together. This knowledge can be used to recommend additional products to customers when they buy a product. In literature, this specific problem is also referred to as frequent itemset mining [1,14].

Clustering
Clustering is the task of grouping (data) items into homogeneous classes. These classes of items that are similar in some ways are called clusters. For the process of clustering, certain properties of the items are taken into account. An example of the application of clustering is customer segmentation. The goal of customer segmentation is to divide a company's customers into groups of individuals that are similar in specific ways relevant to marketing. The properties taken into account could be, for example, age and sex of the customer [1,14].

Classification
Classification is about assigning (data) items to groups. Unlike clustering, the groups are predefined and are not found during the process. That is the reason classification is considered as an instance of supervised learning. This means that a training set of correctly identified observations is available. A sample application domain is pattern matching. With the appropriate knowledge, for example, it is possible to classify e-mails as spam [1,14].

Outlier Detection
An outlier is a (data) element that is significantly different from the rest of the data. The task of outlier detection is to find such anomalies in data. For example, outlier detection is used in the detection of credit card frauds, since the transaction data in a fraud is significantly different from that in normal behavior. Outlier detection can be unsupervised or supervised [1,14].

For each of the above problems, there are a variety of algorithms, such as e.g. the famous k-means algorithm for clustering or the k-nearest neighbor algorithm for classification. However, this article does not aim to elaborate on specific algorithms.

4.2 Mining Data Streams

As mentioned in the previous section, the process of data mining can be applied to many different data sources. These include data streams. Data mining on streams is considered as data stream mining.

The problem is that many conventional data mining algorithms can not handle the constraints of streams. This is because these algorithms have been designed for traditional data and therefore assume random access to the data and potentially infinite processing time. Data streams do not offer these conditions. Because of this, various techniques exist for adapting algorithms to data streams. According to Gaber et al. [10], these techniques can be divided into data-based and task-based techniques:

Data-based techniques

One way to deal with massive data streams is to reduce the incoming data to an amount that the algorithm can handle. This is the goal of the data-based techniques [10].

Sampling

An old established statistical method is sampling. The idea is to select probabilistically whether a data item is processed or not. A well-known instance is the reservoir sampling, where each element has a certain probability that it will be included in the sample. One of the challenges of sampling on streams is that due to the potentially infinite length, the calculation of error bounds for certain sampling rates is not trivial [1,3,10].

Load Shedding

Load shedding refers to the process of dropping sequences of data items from a stream. The problems of load shedding are very similar to those of sampling. It is hard to decide which and how much data can be disregarded. Dropping the wrong data can be quite critical if, for example, during the outlier detection an outlier is ignored [3,10].

Sketching

Another approach is to randomly project a subset of features, which corresponds to vertically sampling the incoming stream. This method works well to compare data streams but is less relevant in the context of data stream mining [3,10].

Synopsis Data Structures

The creation of synopsis data structures refers to the process of applying summary techniques to the incoming data stream. The resulting structures can be used for further analysis. An example for the creation of synopsis data structures is wavelet analysis, where the incoming data

stream is converted into a wavelet by a Fourier transformation. It is possible that in the creation of the synopsis data structures characteristics of the data are lost, which is why in these cases approximate results must be expected. Aggarwal et al. [1] present synopsis data structures as the basis of almost every data stream mining application [1,10].

Aggregation

Another technique that has evolved with the advent of data streams is aggregation. Aggregation refers to the process of calculating statistical measures (e.g., mean or variance) that summarize the incoming data stream. Furthermore, these measurements can be used by the mining algorithms. A concrete example of aggregation are cluster features which is a method to enable clustering on streams. The idea is to keep statistical variables for each cluster which are updated iteratively. Algorithms using an extension of this technique (micro-clusters) are, e.g. CluStream (see Aggarwal et al. [2]) and DenStream (see Cao et al. [6]) [1,10,22].

Task-based techniques

Another way to adapt algorithms to data streams is to change not the data but the algorithms themselves. These approaches are referred as task-based techniques [10].

Approximation algorithms

One approach is to use approximation algorithms, as is done in other areas when hard problems need to be solved. In this approach, mining algorithms are seen as hard computational problems and an inaccurate result is accepted, which can be calculated faster [3,10].

Sliding Window

One technique that has been briefly presented in the context of CEP is the use of sliding windows. The idea is to restrict the considered data elements to the last recently occurred ones. Algorithms using sliding time windows are e.g.. the STREAM algorithm (see [20]) for clustering on data streams or the Online Information Network (see [17]) for classification on streams [10,11].

Algorithm Output Granularity

Algorithm Output Granularity (AOG) is an approach that can respond to limited resources and fluctuating data rates. The idea is that the mining algorithm is designed so that its output can be throttled when it runs out of memory. Then the results are merged until enough memory is available again. An algorithm based on this technique is the LWClass Algorithm (see Gaber et al. [11]) [10,11].

The above list shows that there are many different techniques to enable computing on streams. However, there is not the one technique that enables mining on data streams. Depending on which data mining problem has to be solved and what the concrete application case looks like, variations of the above techniques have to be combined. In addition, the concrete algorithmic techniques have not yet been discussed. For example, the list above does not show how an

algorithm can react to a concept drift or the massive domain constraint. More details can be found in the survey papers by Gaber et al. [11] and Silva et al. [22]. The listed examples for algorithms were also taken from mentioned papers.

In summary, data stream mining is an open research field that has emerged through the advances in hardware and software and is still evolving [8].

5 Comparison

After the previous sections have introduced CEP and data stream mining separately, the following section will explicitly compare the two techniques.

Table 1. Comparison of CEP and data stream mining

Complex event processing	Data stream mining
For processing massive event streams	For processing massive data streams
Rule-based	Data analysis/Data Mining
Pattern matching (matching predefined structures)	Pattern and knowledge discovery (discovering unknown structures)
Follows the sense-process-respond cycle	Results need further processing to react
...	...

Table 1 shows some of the major characteristics of CEP and data stream mining discussed in this article. First of all, we can say that both techniques are used to process massive data streams. In the context of CEP, the concept event abstracts from the raw data elements, but under the hood event streams are the same as data streams. This is also the reason why both techniques are using the methods for processing streams presented in Sect. 4.2. In literature, however, the techniques used in CEP are limited to sliding windows.

Although both techniques deal with data streams, they have their origins in quite different fields of computer science. While CEP has evolved from rule-based systems, data stream mining has evolved from classical data analysis through advances in hard and software. Due to the rapidly growing RAM sizes, the operations that can be performed in-memory are also increasing. As a consequence, CEP and in particular data stream mining become more and more practicable.

The main difference between CEP and data stream mining is that the former is about simple matching of predefined structures on sliding windows and the latter is about finding new patterns and knowledge on data streams. In some cases, however, the tasks of the two techniques are very similar. For example, if spam mails had to be detected, both CEP and data stream mining (classification or an outlier detection) can be used. Specifically, the supervised learning methods come very close to CEP and its applications, since there are also predefined structures used as a knowledge base. The difference is how this knowledge base

is used. For CEP, a pattern can match or not, whereas for pattern matching by classification it is fuzzier. Incoming data does not have to match exactly. Data can be similar to a pattern in a certain way. The outlier analysis can also enable the detection of spam mails. But again, it's not about whether a pattern matches or not, but whether the incoming data elements are significantly different from the rest of the data. In CEP, the pattern that is found must be expressed explicitly by rules and event models, while data stream mining can also find structures that have not been explicitly modeled. As the clustering shows, the areas of data stream mining and CEP can also be very different. Clustering is about finding new structures, which is not possible with CEP. An interesting comparison would be CEP and the association pattern mining on streams. With CEP it is quite possible to carry out an association analysis by explicitly modeling the possible combinations and then counting the triggered events. With a larger number of combinations, however, this becomes impractical very quickly.

Another important difference between CEP and data stream mining is that CEP follows the sense-process-respond circle. Therefore, when a complex event occurs, there is always an action (response). This is because complex event processing was originally intended to control event-driven architectures (EDA) (see Luckham [18]). Data stream mining applications are natively non-reactive. Results of the data mining need further processing to react to situations. As the last row of the table shows, there are other similarities and differences that can not be considered further in the context of this article. Another interesting aspect, e.g., is the fact that there are only a few scientific papers on handling concept drifts with CEP, although this characteristic of streams occurs there. Because the rules in a CEP system are very static, CEP systems can be combined with other techniques to change the rules at run-time (see e.g. [19] or [21]). This could also be a possible approach to dealing with a concept drift. For example, the combination of CEP and data stream mining is possible. In this case, data stream mining finds new patterns and knowledge, which is fed into the CEP system in the form of rules and event models.

In summary, CEP and data stream mining differ in many ways. Surprisingly, there are many overlaps in the fields of application. For CEP, patterns can either match or not, while the matching for data stream mining is fuzzy. Therefore, comparing data stream mining to CEP, knowledge does not always have to be explicitly modeled.

6 Conclusion and Future Work

In this article, we first looked at the special characteristics of data streams and compared them to traditional data. Therefore, CEP was presented as a rule-based method for processing massive data streams. It has been determined that CEP can only be applied if the complex events to be found are defined a priori in the form of rules and event models. However, CEP (alone) does not cover the case if the patterns are unknown. As a solution, the technique of data stream mining

was introduced. It has been shown that it is data mining on streams, which is why the fundamentals of data mining have been introduced. In the following sections it was discussed that data mining techniques were originally designed for traditional data and can not handle the special characteristics of streams without adjustments. For this purpose, some techniques for the processing of data streams were presented and enriched with example algorithms from survey papers. In the end there was a comparison of CEP and data stream mining, which, in addition to many differences, also revealed overlaps in the areas of application.

To summarize the results of this article as a recommendation, CEP should be used when it is necessary to react directly to complex events that can be explicitly modeled before. Data stream mining should be used to discover previously unknown patterns and knowledge from data streams.

While this paper gives an overview of the two techniques and aims to show the origins and differences of both, further papers will deal with more recent findings from the two research fields. One part of our future work will be the development of a decision framework that allows a user to determine the right technique (or the right combination of these techniques) for a specific application.

Acknowledgements. Irina Astrova's work was supported by the Estonian Ministry of Education and Research institutional research grant IUT33-13.

References

1. Aggarwal, C.C.: Data Mining. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-14142-8
2. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29, pp. 81–92. VLDB Endowment (2003)
3. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 1–16. ACM (2002)
4. Bruns, R., Dunkel, J.: Event-Driven Architecture: Softwarearchitektur fürereignisgesteuerte Geschäftsprozesse. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-02439-9
5. Bruns, R., Dunkel, J.: Complex Event Processing. Springer, Wiesbaden (2015). https://doi.org/10.1007/978-3-658-09899-5
6. Cao, F., Estert, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 SIAM International Conference on Data Mining, pp. 328–339. SIAM (2006)
7. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Mobile Netw. Appl. **19**(2), 171–209 (2014)
8. Domingos, P.M., Hulten, G.: Catching up with the data: research issues in mining data streams. In: DMKD (2001)
9. Etzion, O., Niblett, P., Luckham, D.C.: Event Processing in Action. Manning, Greenwich (2011)

10. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. ACM Sigmod Rec. **34**(2), 18–26 (2005)
11. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: A survey of classification methods in data streams. In: Aggarwal, C.C. (ed.) Data Streams. Advances in Database Systems, vol. 31, pp. 39–59. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-47534-9_3
12. Gama, J.: Knowledge Discovery from Data Streams. Chapman and Hall/CRC (2010)
13. Gama, J., Gaber, M.M.: Learning from Data Streams: Processing Techniques in Sensor Networks. Springer, Heidelberg (2007). https://doi.org/10.1007/3-540-73679-4
14. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. Elsevier, Amsterdam (2011)
15. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: Computing on data streams. External Memory Algorithms **50**, 107–118 (1998)
16. Hilbert, M., López, P.: The world's technological capacity to store, communicate, and compute information. Science **332**(6025), 60–65 (2011)
17. Last, M.: Online classification of nonstationary data streams. Intell. Data Anal. **6**(2), 129–147 (2002)
18. Luckham, D.: The Power of Events, vol. 204. Addison-Wesley, Reading (2002)
19. Mehdiyev, N., Krumeich, J., Enke, D., Werth, D., Loos, P.: Determination of rule patterns in complex event processing using machine learning techniques. Procedia Comput. Sci. **61**, 395–401 (2015)
20. O'callaghan, L., Mishra, N., Meyerson, A., Guha, S., Motwani, R.: Streaming-data algorithms for high-quality clustering. In: Proceedings 18th International Conference on Data Engineering, pp. 685–694. IEEE (2002)
21. Pielmeier, J., Braunreuther, S., Reinhart, G.: Approach for defining rules in the context of complex event processing. Procedia CIRP **67**, 8–12 (2018)
22. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., De Carvalho, A.C., Gama, J.: Data stream clustering: a survey. ACM Compu. Surv. (CSUR) **46**(1), 13 (2013)