Predictive Complex Event Processing: A conceptual framework for combining Complex Event Processing and Predictive Analytics

Lajos Jenő Fülöp
University of Szeged
Árpád tér 2. H-6720
Szeged, Hungary
flajos@inf.u-szeged.hu

Gabriella Tóth
FrontEndART Software Ltd.
3. I/5. Zászló str., H-6722
Szeged, Hungary
gtoth@frontendart.com

László Vidács
RGAI, University of Szeged &
Hungarian Academy of
Sciences
Árpád tér 2. H-6720
Szeged, Hungary
lac@inf.u-szeged.hu

Árpád Beszédes
University of Szeged
Árpád tér 2. H-6720
Szeged, Hungary
beszedes@inf.u-
szeged.hu

Hunor Demeter
Nokia Siemens Network
Hungary
hunor.demeter@nsn.com

Lóránt Farkas
Nokia Siemens Network
Hungary
lorant.farkas@nsn.com

ABSTRACT

Complex Event Processing deals with the detection of complex events based on rules and patterns defined by domain experts. Many complex events require real-time detection in order to have enough time for appropriate reactions. However, there are several events (e.g. credit card fraud) that should be prevented proactively before they occur, not just responded after they happened. In this paper, we briefly describe Complex Event Processing (CEP) and Predictive Analytics (PA). Afterwards, we focus on a major future direction of CEP, namely the inclusion of PA technologies into CEP tools and applications. Involving PA opens a wide range of possibilities in several application fields. However, we have observed that only few solutions apply PA techniques. In this paper, we define a conceptual framework which combines CEP and PA and which can be the basis of generic design pattern in the future. The conceptual framework is demonstrated in a proof–of–concept experiment. Finally we provide the results and lessons learned.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems; I.5.4 [**Pattern Recognition**]: Applications

General Terms

Design, Experimentation

Keywords

CEP, Complex event processing, predictive analytics

1. INTRODUCTION

Complex Event Processing (CEP) deals with collecting events from multiple sources, detecting patterns, filtering, transforming, correlating and aggregating them into complex events. Examples of complex events are: the 2009 Stock Market Crash (finance and banking), the 2007 DOS attack of the Estonian IP network (security) and the churn rate increase at a Communication Service Provider (telecom). Event processing is not a new concept; it is widely used in the industry to solve problems (e.g. a substantial part of the GSM and internet infrastructure processes events). Typically each problem domain will define its own event formats and communication protocols. Understanding events semantics and relationship requires deep domain expertise. CEP alone cannot solve this problem, however, it can provide a platform, where all event sources can be plugged in, the events can be transformed to a normalized form, and event operators can be applied on them to produce complex events. It is important to understand that the value of the complex events decreases with time (e.g. tsunami warning issued in 10 minutes vs. in 10 hours). This imposes several technical requirements on CEP platforms: high scalability, high computation power and low latency.

An other research area is Predictive Analytics (PA) that deals with the analyzation of historical data to give predictions about a future event. For the prediction, PA applies several statistical and data mining techniques, for example clustering, classification, regression and so on. By applying these techniques, PA builds *predictive models* which represents certain circumstances between available features or predictors related to the event. PA faces problems like how to define predictors and how to calculate them, how to define the event, and so on.

Synergy between CEP and PA offers valuable chances. PA deals with every kind of prediction (long term, short term, classification, regression, etc.), while CEP deals with detecting complex events occurring in real time. By combining these two areas both can support the other: (i) PA supports CEP because one could realize

complex events in the short term future, while (ii) CEP techniques can ensure the calculation of predictors for PA.

The paper is organized as follows. In the next section a brief introduction to these two areas are presented, followed by a section on related work. In Section 4 we present the conceptual framework for combining CEP and PA and it is demonstrated through a proof-of-concept experiment. Next, we discuss threats to validity and in the final section we draw conclusions.

2. BACKGROUND

Complex event processing. By definition an event is *"anything that happens, or is contemplated as happening"* [5]. The events can occur in the real world (e.g., an airplane has landed) or can be virtual (e.g., an airplane has landed in an airplane simulator). Physically, events can come from an external database, an RFID data sensor, a service, an enterprise information system, etc. [16]. Events are distinguished based on their complexity: an event can be either simple or complex. A complex event is the abstraction of simple or complex events, for example, the landing of an airplane is a complex event which is built up of several simple or complex events: the pilot decreases the trust, increases the drag, controls the crosswind, and introduces flare. The events can be linearly ordered in event streams or partially ordered in event clouds. The events travel from the *event sources or producers* through *event channels* to the *event sinks or consumers*. An event processing engine acts as event sink for simple events, and as event source of complex events. While the events travel from sources to sinks, various Event Processing Agents (EPAs) can perform computation on them. There are different types of EPAs: simple event processing EPA (e.g. filter and routing), mediated event processing EPA (e.g. enrichment, transformation and validation), complex event processing EPA (e.g. pattern detection) and intelligent event processing EPA (e.g. decisions). The EPAs can be connected to each other to form an Event Processing Network (EPN).

Predictive analytics. Predictive analytics determine predictive models to exploit patterns found in historical and transactional data and identify risks and opportunities. Models capture relationships among many factors to allow assessment of risk or potential associated with a particular set of conditions, guiding decision making for candidate transactions. Predictive analytics is used in several fields: financial services, insurance, telecommunication, retail, travel, healthcare, pharmaceutical industry, etc. Predictive analytics methods based on complex event data can make predictions about some attributes of the monitored system based on the previously monitored events. Generally a prediction process consists of four steps: (1) collect and preprocess raw data; (2) transform preprocessed data into a form that can be easily handled by the (selected) machine learning method; (3) create the learning model (training) using the transformed data; (4) report predictions using the previously created learning model. By using recent data based on the learning model trained for the previously monitored events, future events can be predicted.

2.1 Comparison of CEP and PA

Both CEP and PA target the problems related to processing large amount of runtime data of large software systems, usually collected from execution traces, in order to detect undesired behavior (like runtime failures or performance degradation) or other specific behavior patterns of interest. With these technologies, runtime data is usually processed online, and the decision is made based on current or past data, while sometimes the goal is the prediction of future events. In order to apply CEP and PA together, we compare these technologies by investigating their differences and common parts.

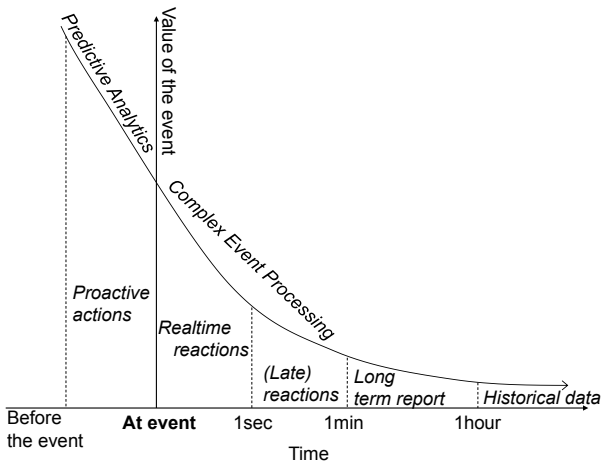

Figure 1: Value of the knowledge about the event

Aim. The value of complex events decreases over time, as already mentioned through the example of the tsunami warning in the introductory section. A warning issued as earlier as possible in time is an important factor in CEP, but considering the timeline of an event, a warning issued before the event itself is even more valuable. Figure 1 shows the value of the information about the event. The x axis represents the time while y represents the value of the information about the event. Knowledge about the event before it happens is an advantage, proactive actions can performed to precede the undesired event. On the other side, Complex Event Processing detects the event after it has happened and only posterior reactions can performed. However, several situations require exact knowledge about the event. In these cases, PA cannot be used because its uncertainty.

Automatization. The other most notable difference lies with the fact that complex event processing engines require predefined rules or patterns to calculate a complex event, which usually has to be designed and deployed to the system manually, while the goal of predictive analytics is to automate these processes, build rules and patterns automatically. Based on this, there is an assumption that rule or pattern developers have the required preliminary knowledge, which sometimes is not available (in the case of unknown patterns) or not so precise. The manual setting of the rules and patterns are in a sense a weak point of complex event processing, which can readily be supported by the techniques of predictive analytics. Predictive analytics is not a key part of most complex event processing engines, but current CEP engines can be extended with PA.

Application. The application timeframe when the event data is processed is different in the two cases also. In the case of PA, the learning phase takes place much before the usage of the model, and its techniques require certain computations in advance. On the other hand, CEP techniques are most often applied in real-time.

3. RELATED WORK

Muthusamy et. al. [6] presented a work about predicting a subscription in the future in publish/subscribe context. They applied Markov model for learning and for prediction. In the presented application it predicts the probability of that a given subscription will occur in the future. They achieve good results, e.g. high precision. An important experience was, that the machine learning model out-

performs a hand-crafted model defined by domain expertise.

There is a proposal mechanism for automating both the initial definition of rules and the update of rules over time [12]. Henriques et. al. [2] described their system, HOLMES. The system among several things applies some machine learning capabilities as well. They used machine learning in order to detect anomalies in time series but in an unsupervised context.

In the industry, TIBCO Software Inc. has ambition to use event processing technologies for prediction. They developed not only event processing tool (TIBCO BusinessEvents) but predictive analytics tool (TIBCO Spotfire Miner) as well, and they introduced predictive business (TIBCO Predictive Business) [9] which identifies the patterns in historical events, projecting them into the present and future. Ammon et al. [8] connected the CEP to business activity monitoring and business process management. Business process management and real-time business activity monitoring are newly discussed as the preconditions for a so-called predictive business and the competitiveness of enterprises in the future.

Since the known event patterns can be derived from heuristics (e.g., from business activity monitoring view), the unknown event patterns cannot. According to Widder et al.[14], unknown event patterns can be found with the help of event processing agents (EPAs) by analyzing the event cloud of an organization and using specific algorithm to detect them. If an EPA detects an unknown pattern – which seems to be a suspicious event combination – determined by discriminant analysis, EPA sends an alert and this pattern is saved in the database. Discriminant analysis analyzes multidimensional data to discover relationship between data.

Previously, we published a poster about how to connect CEP and PA [11], and a technical report [3] about CEP and PA.

4. CONCEPTUAL CEP - PA FRAMEWORK

CEP detects complex events in an event cloud. The basic computational unit is an Event Processing Agent (EPA) that performs a predefined task on incoming events (e.g. filtering, aggregating, detecting patterns etc.). EPAs can interconnect using event channels to form Event Processing Networks (EPN). EPNs produce complex events as output. We call the most important event as the *primary complex event*. With a joint CEP and PA framework this primary complex event can be predicted. The conceptual framework of the interworking CEP and PA solution is illustrated in Figure 2. The conceptual framework works on point-based events but it could be adapted to interval based events as well. An ellipse denotes event source or event sink, while a box represents an information processor component and a line between the boxes denotes the information flow. The boxes and lines drawn with straight lines are the parts of a generic CEP application, while the boxes and lines drawn with dashed lines represent the introduced PA components.

4.1 Requirements of the framework

Let us suppose, that we have a generic CEP application, represented by the boxes drawn with straight lines in Figure 2. In this system, raw events comes from *event sources* and processed by the *Event Processing Network*. Then, EPN detects the *Primary Complex Event* (PCE) and sent it to the registered *event sink* which triggers reactions when the event occurs, e.g. notifies the business actor or triggers an automatic reaction.

Our motivation for the CEP–PA synergy is that in many cases the

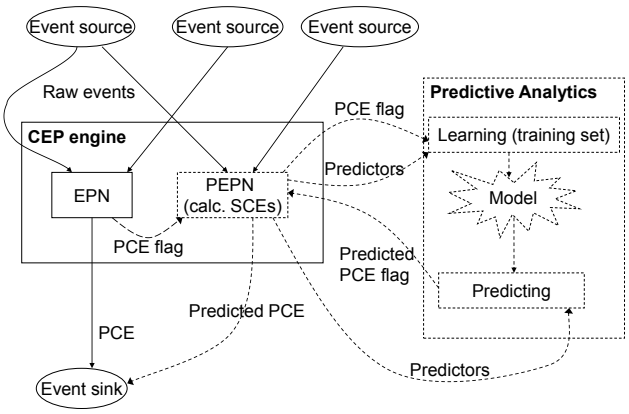

Figure 2: CEP-PA conceptual framework

reaction should be triggered earlier, even before the complex event occurs. In these cases, events can be predicted by PA models incorporated to the CEP solution. The initial CEP problem can be extended with the PA component in several ways but first of all we define requirements for the synergy:

- PA has to give the prediction for the CEP application about future occurrences of PCE.
- CEP should ensure the prediction target (PCE) and the predictors for PA. The predictors are the key performance indicators of the complex events. Note, that the prediction may require additional sources. For example, different sources required to detect or to predict rain.
- synergy of CEP and PA should be transparent: the generic CEP application should not be modified and its interfaces should not be affected.
- synergy should not affect the maintainability of the CEP engine. The original cohesion, complexity and coupling of the CEP engine should be affected as small extent as possible.

The first and second requirements can fulfilled easily. The third requirement demands for the same event source with the same raw input events, and the same event sink with the same (output) complex event as in the generic CEP application has. In a conceptual framework there are two ways to match this requirement: (1) by introducing *Predictive Event Processing Agents* (PEPA) in the event processing network that can deal with the synergy of CEP and PA and (2) by introducing separate *Predictive Event Processing Network(s)* (PEPN) that can deal with the synergy of CEP and PA.

Both techniques match the first, second and third requirements but only the second approach fits the fourth requirement as well. By introducing PEPA-s the maintainability of the CEP application decreases because the PEPAs decrease the cohesion and increase the number of connections (by this way the coupling increases as well). Therefore the best solution is to introduce a separate and specific *Predictive Event Processing Network* (PEPN).

4.2 Synergy of CEP and PA

The boxes drawn with dashed lines in Figure 2 represent the new components required for the PA extension. In the followings, we describe the operation of the conceptual framework.

First, *raw events* come from the event sources. Afterwards, *EPN* tries to detect *PCE* and send it to the *event sink*. At the same time, EPN sends the *PCE flag* (PCEF) to the *PEPN* about whether PCE occurs or not. Next, *PEPN* extends the *training set* of PA with previously calculated predictors and with the *PCEF*. At the same time, *PEPN* also sends the currently calculated predictors to PA and then PA gives a prediction about the PCEF (Predicted PCEF – PPCEF) based on the current learning model and on the predictors. Finally, *PEPN* sends the predicted event to the *event sink*.

The PA component contains a machine learning part which works on the *continuously* extended training set. A large set of predictors is necessary for building the internal machine learning model. In the conceptual framework, the *PEPN* itself produces the predictors (by this way fulfilling the second requirement). The PEPN processes raw events and produces secondary complex events (i.e. predictors for PA). Another requirement for PA is the target of the prediction, which is the *Primary Complex Event Flag* (PCEF).

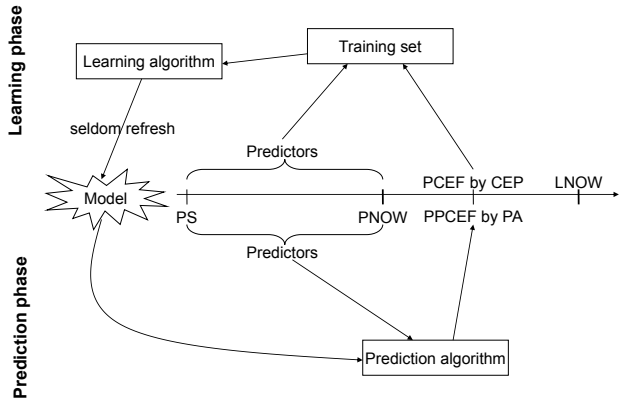

Figure 3: CEP-PA events illustrated chronologically

Figure 3 demonstrates the working of the conceptual framework in time. The *x* axis represents time, with measurement points. The upper part of Figure 3 is a snapshot of the learning phase (made at *LNOW* point), while its lower part is a snapshot of the prediction phase (made at *PNOW*) of the conceptual framework.

During the learning phase, the training set is continuously extended with predictors and with PCEF. Afterwards, the learning model is refreshed. Because the model refreshment is a resource consuming task therefore it should be performed seldom and on a separate thread during the implementation. During the prediction phase, a prediction (Predicted PCE flag – PPCEF) is given based on the current model and on the predictors. On Figure 3 we have the following intervals and points:

- *predictors* interval: predictors are calculated between *predictors' startpoint* (PS) and *Prediction phase Now point* (PNOW). For the learning phase, these predictors represent the *preceding* context before the PCE occurs. For the prediction phase, these predictors represent the basis of the prediction.

- *PCEF* (PCE flag) is a boolean variable, it represents whether the PCE is occured or not. It is calculated from the PCEF time point or from an interval ended at PCEF time point (by this way the conceptual framework supports both point or interval based events)

- *predicted PCE flag* (PPCEF) represents the forecasted PCE.

PCEF and *PPCEF* represent the same time but in different context. *PCEF* represents historical data during the learning phase, which is used to extend the training set. *PPCEF* represent the unknown future and the prediction about the future.

In the latter part of the paper we will see that every variable takes values based on the absolute distance from *PNOW*. By this way, *PSD* (*PS* distance) is defined as the distance between *PS* and *PNOW*, while *PCEFD* (*PCEF* distance) is defined as the distance between *PNOW* and *PCEF*. For example, a setting could be the following: $PSD = 5, PCEFD = 2$.

Lastly, we note that the predicted event can be used to define further complex events. For example, when depending on an other event we do not want to consider the prediction, then a new complex event may help.

4.3 Proof of concept

The experiments are performed on a CEP program which works on an entry system of a building. Conferences and talks are held in the building, by this way the incoming and outgoing traffic increase randomly. The traffic data about incoming and outgoing persons is supplied by the main entrance of the building in every half an hour. The task of the CEP engine is to listen this traffic data, and in case of too high *incoming* traffic it has to issue an *Entry System Overloaded* (ESO) complex event. ESO fits to the primary complex event (PCE) of the conceptual framework.

The entry system has two *working states*: *low* and *normal*. The *low state* limits the number of entrants, but its maintenance cost is much cheaper than the normal state's one. Normal state has no limitation considering the number of entrants. The default working state is low. When the low state gets overloaded then the entry system has to be switched to the normal state. There are several problems which motivate the detection of the overloaded state. Too many entrants demand opening new entry doors. Opening new entry doors requires relatively long time, because of the hurrying crowd. Too many entrants could cause overload and shutdown in the wireless entry system. The suppliers in the building (e.g. restaurants, clerks, etc.) have to be notified to prepare for the unusually large number of entrants (e.g. a new clerk should be called from his 10th floor office to the ground floor, etc.)

The experiments are performed on the database of a public entry system [13]. The developed CEP–PA proof–of–concept application is based on the conceptual framework presented in Section 4.

4.3.1 The CEP solution

The original database contains information about concrete events (conferences, talks). However, we define a special complex event based on a special rule: when the number of entrants is greater than 25 during the last 1.5 hours, then ESO has happened. Note that the half-hour measurements could be more frequent, e.g. it could be 10 seconds. By this way it does not affect the validity of the experiment, it is adaptable to a real–time problem as well. The presented application is implemented with Esper.

Figure 4 shows the architecture of the conceptual framework's proof–of–concept implementation. In the current implementation, the events for the CEP engine are simulated by the *Entry system simulator* which simply reads the database file [13] and forwards the data to the CEP application. Afterwards, the application detects the ESO based on the rule (number of entrants is greater than 25

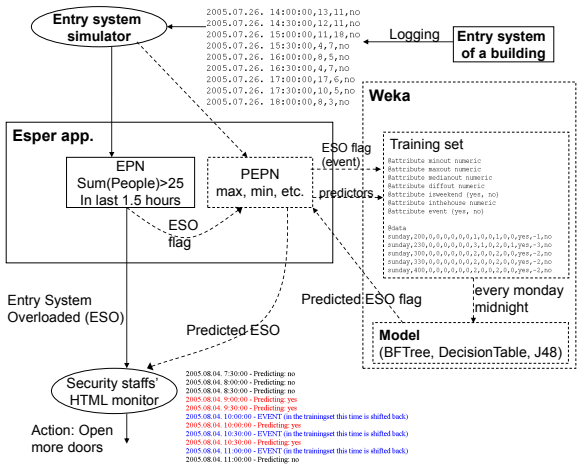

Figure 4: CEP-PA proof of concept

during the last 1.5 hours) and it writes the event into an HTML file (in blue). Implementation of the conceptual framework's EPN item contains a simple query with an event listener. The query selects the sum of the incoming people during the last 1.5 hours (which means an three length interval because of the measurements are performed at every half an hour):

```
select sum(PeopleCount), DateAndTime from
       CalcEventTick.win:length_batch(3)
```

CalcEventTick.PeopleCount represents the number of incoming people while *CalcEventTick.DateAndTime* represents the timestamp. Finally, the event listener checks whether the sum is greater than.

4.3.2 Extension with PA

The prediction is based on predictors that are calculated from the number of entries and exits, and the timestamp of the measurement. So, we have the following raw events: *entries* - the number of incoming persons during the last half an hour; *exits*: the number of outgoing persons during the last half an hour (exits); the *timestamp* of the measurement.

Implementation of the conceptual framework's PEPN item is also really simple. The following five simple (secondary) complex events have been defined and detected with the Esper application [1] to ensure the predictors for PA: (1) the summed number of persons coming in during the *Predictors* interval (see Figure 3); (2) the difference between the entrants at *PS* and *PNOW*; we had presumed that it increases when an event is approaching; (3)the day of the week (e.g. Monday) at *PNOW*; (4) the maximum/minimum/average/median of entries/exits during the *Predictors* interval; (5) the number of people in the building at *PNOW*.

Note, that some predictors cannot be calculated with CEP (e.g. day of the week), predictors like that are determined programatically. During the first week, the synergy cannot provide any forecast, it just extends the training set in the PA framework (in the presented solution we applied Weka [15] internally, Figure 4 shows a fragment of the Weka training set). At the beginning of the second week (at Monday midnight), the first learning model is built, and afterwards the training set is extended continually with the new data, and the learning model is refreshed at every Monday midnight. Finally, the CEP–PA synergy can give forecasts in every half an hour. The forecast is written to the HTML file (in red).

Table 1: Prediction results

Algorithm	PSD	PCEFD	Precision	Recall	F-measure
BFTree	3	1	0,93	0,92	0,93
DecisionTable	3	1	0,91	0,94	0,92
BFTree	5	2	0,94	0,82	0,88
DecisionTable	3	3	0,91	0,83	0,87
DecisionTable	5	2	0,88	0,84	0,86
DecisionTable	8	3	0,92	0,79	0,85
BFTree	3	3	0,96	0,76	0,85
BFTree	8	3	0,94	0,77	0,85
DecisionTable	5	4	0,90	0,79	0,84
DecisionTable	8	5	0,88	0,78	0,83
BFTree	5	4	0,95	0,73	0,83
BFTree	8	5	0,92	0,72	0,81
J48	8	5	0,89	0,72	0,80
J48	5	4	0,85	0,67	0,75
J48	8	3	0,81	0,63	0,71
J48	3	3	0,80	0,63	0,70
J48	5	2	0,72	0,6	0,65
J48	3	1	0,65	0,56	0,60

Experiments and prediction results. We have performed experiments with several settings of the PSD and $PCEFD$ parameters (see Section 4) and with three machine learning algorithms of Weka: (1) BFTree is the best-first decision tree classifier [10]; (2) DecisionTable is a table that describes conditions with actions to perform [4]; (3) J48 is the Java implementation of the well-know C4.5 tree [7].

Values for PSD are 3, 5 and 8 while values for $PCEFD$ are 1, 2, 3, 4 and 5. However, not every combination of the parameters are tested. During the experiments we applied the well–known statistical indicators: precision, recall and f–measure. They are based on the ratio of true positives, false positives and false negatives which are defined as follows. *True Positive (TP)* occurs when PA predicts an event that will occur in the future (correct prediction). *False Positive (FP)* occurs when PA predicts an event that will not occur in the future (false prediction). *False Negative (FN)* occurs when PA not predict an event that will occur in the future (missed prediction).

Based on True Positives and False Positives we can measure the *precision* of the prediction: $\frac{|TP|}{|TP+FP|}$. Based on True Positives and False Negatives we can measure the *recall* of the prediction: $Recall = \frac{|TP|}{|TP|+|FN|}$. *F-measure* is the weighted harmonic mean of precision and recall, so it represents an overall goodness of the prediction: $\frac{2*Precision*Recall}{Precision+Recall}$.

We take into consideration that the PCE/ESO is defined on a three point interval (number of entrants is greater than 25 during the last 1.5 hours while the measurements are taken at every half an hour). Because of that, we accept a prediction as True Positive if the event occurs at a wider range in the future:

$$future = [\max(PNOW + 1, PCEFD - 3), PCEFD]$$

In Table 1 we give the prediction results in case of different parameters and different machine learning algorithms. The results are ordered by F-measure. Considering the algorithms, the DecisionTable and BFTree algorithms perform similarly, while J48 performs really bad compared to the other two. The best choice for the PSD parameter is 3 while the best choice for the $PCEFD$ parameter is 1 or 2. However, other parameter values perform good results as well.

4.4 Discussion

The presented proof-of-concept is a simplified solution and it may seems to not describe well the real world situation. In the followings, we show points against the demonstration presented previously, and we explain why these points do not decrease significantly the validity of the presented proof of concept:

- In certain aspect, the original problem is not CEP related because the measurement is taken in every half an hours, which cannot be named as a real-time problem. Irrespectively of this fact, it was applicable as a *demonstration* (and not for a case study), because the timeframe could be set smaller in an other scenario, e.g. one second, making it a real-time PoC.

- Synergy with PA could cause performance problems? In the PA framework the resource consuming task was only the model refreshment (about 3-4 seconds), but it did not affect the real-time work, because the model was refreshed *periodically*, not in every measurement point.

- *"If you react before the complex event happens, you kill the training set."* This is a real threat but we predict the event because we want to make some proactive and preventive actions before it happens. On the other side, after the prediction it is difficult to determine that the event would have happened if the proactive actions had not performed previously. However, this will not a problem if the event detection does not depend on the proactive actions (for example, it is true for the previously demonstrated proof–of–concept).

5. CONCLUSIONS

In this paper we investigated an open research direction of Complex Event Processing: *"the advantages of CEP can be enhanced with the help of predictive analytics?"*. We elaborated this question in greater details, which is the major result of this work. We presented a conceptual framework for incorporating Predictive Analytics into Complex Event Processing, and afterwards the framework was demonstrated through experiments performed on real-world data using a proof-of-concept. The achieved results are promising, the CEP solution was extended with predictive capabilities and most of the events were predicted successfully.

Furthermore, we also plan to make a case study on a large, real-world CEP system because the presented solution for the synergy is only validated on a proof-of-concept example. A real-world CEP system defines more complex patterns, solves more complex problems and requires more complicated predictors than the presented proof-of-concept.

6. ACKNOWLEDGEMENT

This research was supported by the Hungarian national grant GOP-111-11-2011-0038.

7. ADDITIONAL AUTHORS

Additional authors: Tibor Gyimóthy (University of Szeged, email: gyimothy@inf.u-szeged.hu), Gergő Balogh (University of Szeged, email: geryxyz@inf.u-szeged.hu)

8. REFERENCES

[1] Homepage of Esper/NEsper. http://www.espertech.com/.

[2] P. H. dos Santos Teixeira, R. G. Clemente, R. A. Kaiser, and D. A. V. Jr. Holmes: An event-driven solution to monitor data centers through continuous queries and machine learning. In *DEBS '10: Proceedings of The 4th ACM International Conference on Distributed Event-Based Systems*, pages 216–221. ACM, 2010.

[3] L. J. Fülöp, G. Tóth, R. Rácz, J. Pánczél, T. Gergely, A. Beszédes, and L. Farkas. Survey on Complex Event Processing and Predictive Analytics. Technical report, University of Szeged, 2010.

[4] R. Kohavi. The power of decision tables. In *ECML '95: Proceedings of the 8th European Conference on Machine Learning*, pages 174–189, London, UK, 1995. Springer-Verlag.

[5] D. Luckham and R. Schulte, editors. *Event Processing Glossary – Version 1.1*. Event Processing Technical Society, 2008. URL: http://www.ep-ts.com/component/option,com_docman/task,doc_download/gid,66/Itemid,84/.

[6] V. Muthusamy, H. Liu, and H.-A. Jacobsen. Predictive publish/subscribe matching. In *DEBS '10: Proceedings of The 4th ACM International Conference on Distributed Event-Based Systems*, pages 14–25. ACM, 2010.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[8] C. S. Rainer v. Ammon and C. Wolff. Domain specific reference models for event patterns for faster developing of business activity monitoring applications. 2007. URL: http://www.citt-online.com/downloads/ReferenceModelsEventPatterns_with_Appendix_v3.pdf.

[9] V. Ranadive. *The Power to Predict: How Real Time Businesses Anticipate Customer Needs, Create Opportunities, and Beat the Competition*. McGraw-Hill Pub. Co., 2005.

[10] H. Shi. Best-first decision tree learning. Technical report, University of Waikato, 2007.

[11] G. Tóth, L. J. Fülöp, L. Vidács, A. Beszédes, H. Demeter, L. Farkas, and T. Gyimóthy. Complex event processing synergies with predictive analytics. In *DEBS '10: Proceedings of The 4th ACM International Conference on Distributed Event-Based Systems*, pages 95–96. ACM, 2010.

[12] Y. Turchin, A. Gal, and S. Wasserkrug. Tuning complex event processing rules using the prediction-correction paradigm. In *DEBS '09: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, pages 1–12, New York, NY, USA, 2009. ACM.

[13] UC Irvine Machine Learning Repository - CalIt2 building people counts. http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/.

[14] A. Widder, R. v. Ammon, P. Schaeffer, and C. Wolff. Identification of suspicious, unknown event patterns in an event cloud. In *DEBS '07: Proceedings of the 2007 International Conference on Distributed Event-Based Systems*, pages 164–170, New York, NY, USA, 2007. ACM.

[15] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

[16] C. Zang and Y. Fan. Complex event processing in enterprise information systems based on rfid. *Enterp. Inf. Syst.*, 1(1):3–23, 2007.