F. Terroso-Sáenz et al., Int. J. of Design & Nature and Ecodynamics. Vol. 11, No. 3 (2016) 275–283

TOWARDS ANTICIPATE DETECTION OF COMPLEX EVENT PROCESSING RULES WITH PROBABILISTIC MODELLING

FERNANDO TERROSO-SÁENZ, AURORA GONZÁLEZ-VIDAL & ANTONIO F. SKARMETA
Department of Information and Communications Engineering, Computer Science Faculty, University of Murcia, Spain.

ABSTRACT
Nowadays, Big Data implies not only the need of processing high volume of data, but also do it in a timely manner. In this scope, the Complex Event Processing (CEP) paradigm has arisen as a prominent real-time rule-based solution. Due to its reactive nature, a CEP system might suffer from slight delays in the activation of its rules that could not be desirable in certain environments. As a result, the present work introduces a novel mechanism that intends to anticipate the activation of event-based rules and, thus, come up with even faster CEP systems. This is achieved by means of a probabilistic modelling of each rule's precondition. Finally, the proposal includes a preliminary evaluation so as to show its suitability.
Keywords: *complex event processing, event processing rules, predictive analysis.*

1 INTRODUCTION

In this day and age, the amount of data produced in our world during the last years has experienced exponential growth, reaching 2.5 exabytes of data per day [1]. In this frame, the term "Big Data" stands for the collection of solutions for managing and extracting meaningful information from this data. This is done by providing an alternative to traditional database management systems and data-mining techniques. In that sense, big data have been characterized by the three V's [2], *volume*, *variety* and *velocity*.

Concerning the velocity dimension, that is the speed at which the incoming data must be processed, there is an increasing necessity to do that in nearly real time. As a matter of fact, a road-traffic management system, which might receive thousands of new mobility reports every single instant, would be undoubtedly interested in the detection of traffic jams as soon as they occur so as to perform the most appropriate actions to counteract the congestions in a rapid manner.

In this scope, Complex Event Processing (CEP) is a relatively novel and promising software technology that aims to address the development of recognition systems [3]. The goal of CEP system is the timely detection of situations of special significance within an organization or domain given streams of low-level information items, so-called events, that cannot be directly handled by humans. To do so, a CEP system consists of a palette of asynchronously interconnected Event Processing Rules (EPRs). Such rules are defined by expert knowledge indicating their pre-conditions and the actions to be taken if they are fulfilled.

Despite its real-time processing capabilities, one of the major drawbacks of CEP is its reactive nature as an EPR is only activated when its conditions are fully satisfied. However, the actual value of an event derived by an EPR activation decreases with time. In order to cope with this limitation, a prominent approach has made use of predictive analysis in order to forecast the occurrence of

WIT CONFERENCES
This paper is part of the Proceedings of the International Conference on Big Data (Big Data 2016)
www.witconferences.com

ISSN: 1755-7437 (paper format), ISSN: 1755-7445 (online), http://www.witpress.com/journals
DOI: 10.2495/DNE-V11-N3-275-283

certain events in advance [4–6]. Nonetheless, this is achieved by means of external tools decoupled of the CEP system.

In this scope, the present work presents a novel approach to provide predictive features to the CEP paradigm. In particular, the introduced solution focuses on anticipating the activation of a rule and, thus, make up its outbound derived event in advance. To do so, for each EPR, the proposed mechanism analyses its list of pre-defined requirements to fire. Then, it incrementally generates an ad-hoc predictive model of such requirements at the same time the system receives the incoming raw events, so it does not rely on a preliminary data collection step. Finally, this predictive model determines when the EPR can be activated in advance even though its predefined requirements have not been fully accomplished yet.

All in all, the mechanism that we introduce here is a completely different approach regarding previous solutions which focus on generating future events by means of predictive-analysis tools decoupled of the system's EPRs. In other words, while previous approaches provide solutions of how a CEP system can deal with externally-predicted events, the present work centers on how to generate such events within a CEP system. Consequently, this work intends to enrich the CEP paradigm with actual pro-active capabilities as a built-in feature by automatically transforming regular reactive EPRs into proactive ones.

The remainder of the paper is structured as follows: an overview of the state of the art of predictive analysis in the CEP domain is put forward in section 2. A detailed explanation of the proposed predictive mechanism is stated in section 3. Then, section 4 discusses the results of a preliminary experiment. Finally, the main conclusions and the future work are summed up in section 5.

2 BACKGROUND

2.1 Complex event processing

A CEP system endlessly reads *raw* events from a set of event producers [3]. Then, it performs a set of operations on these events so as to give insight into new knowledge in the form of new *derived* events that are delivered to the event consumers. Both raw and derived events are generally characterized by a set of inner fields that contain their main attributes. The entities in charge of performing such operations are the EPRs. Thus, an EPR takes as input one or more event streams. When its condition part is fulfilled, it asynchronously delivers new derived events to other rules through a *push-style* communication. All these connections are enabled by an inner CEP engine.

2.2 Predictive analysis in the CEP domain

In recent years, there are few research efforts which have explored the possibility of combining predictive analytics (PA) methods with CEP to provide proactive solutions. Initially, it was proposed in [5], where the authors presented a conceptual framework for combining PA with CEP to get more value from the data. Although the idea of combining both technologies was encouraging, they did not support their work with any practical application. Another example is given in [6] where the authors used Expectation Maximization (EM) algorithm to detect complex events but its complexity increases exponentially as the training dataset increases. Finally, in [7], the authors provide a basic framework for combining time series prediction with CEP. The authors highlighted the open issues related to prediction components such as model selection and model update as new data arrive but did not address these issues and left it for their future work.

A common feature of all the aforementioned solutions is that they use different PA techniques to generate forecasted events. Next, such events feed a CEP system in charge of detecting situations of interest in advance. Thus, PA is not actually integrated in a CEP solution. In addition to that, the mentioned PA tools need, in most of the cases, a historic dataset to compose their models. In that sense, we should bear in mind that CEP is designed, as a real-time solution, to operate in environments where collecting reliable historical data might be a difficult task. On the contrary, the present work differs from previous solutions as it proposes a predictive mechanism regarding the pre-conditions of the EPRs so that the PA is included within the CEP system. Moreover, the adopted solution allows to incrementally generate the predictive model at the same time the CEP system receives/generates its events. Hence, a preliminary data collection step is no longer required.

3 PROACTIVE EVENT PROCESSING RULES

This section is devoted to explain in detail the proposed mechanism for proactive EPRs.

3.1 Target EPR types

Two types of EPR's are supported by the current version of the solution, the *intersection-based EPRs* and the *sequence-based EPRs*.

3.1.1 Intersection-based EPRs

The first type comprises those rules whose preconditions take the form of a list of events where all of them should occur in any order before the rule files. The following pseudocode shows the general form of this rule type,

```
CONDITION Event 1(cond11, cond12,..., cond1k)
            AND Event 2(cond21, cond22,... ,cond2p)
            ...
            AND Event n(condn1, condn2,..., condnq)
ACTION new DerivedEvent();
```

As we can see, this type of rule generates a new Derived Event when a set of n events `{Event _1, Event_2, ..., Event_n}`, each one accomplishing a set of conditions condxy occur. A example of this type of rule in a road-traffic management system monitoring potentially risky situations might be the following,

```
CONDITION CarCrashEvent as c
            AND HighTrafficDensityEvent(road = c.road)
            AND RainyWeatherEvent(dist(area,c.road) < d_min)
ACTION      new DangerousSituationEvent(c.road);
```

For this type of rule, the goal of the mechanism is to anticipate the generation of the Derived Event so that the rule does not need to wait until receive the n incoming events to fire. On the contrary, it will generate a probabilistic model so as to uncover the smallest subset of m incoming events ($m < n$) that allows to generate in advance the rule outcome.

3.1.2 Sequence-based EPRs

This second type of rule is similar to the previous one, but its pre-conditions defines a particular order among the incoming events by means of the followed by operator ($\rightarrow$). Consequently, the general form of this target EPR is as follows,

```
CONDITION Event 1(cond11, cond12,..., cond1k) ->
          Event 2(cond21, cond22,... ,cond2p) -> ...->
```

```
          Event_n(condn1, condn2,..., condnq)
ACTION    new DerivedEvent();
```

For this type of rule, the goal of the mechanism is to generate a probabilistic model so as to uncover the smallest subsequence of m incoming events {Event_1 → Event_2 → ... → Event_m} ($m < n$) that allows to generate the rule outcome in advance.

3.2 Mechanism's components

Figure 1 shows the proposed solution for proactive EPRs in a CEP system deployment. The following sections describe its three different modules.

3.2.1 EPR syntax analyser

The *EPR syntax analyser* is executed just before the system is launched for execution. Its main goal is to collect, for each *intersection-based* or *sequence-based* rule, (1) its type, (2) the events in its condition part and 3) its generated. derived event. Then, these three items are sent to the *predictive model generator.*

3.2.2 Predictive model generator

This component is the core element of the mechanism as it is in charge of compose the probabilistic model for each EPR. This implies two different steps.

Firstly, the module initializes a different probabilistic graph model for each EPR matched by the *EPR syntax analyser*. Depending on its type, its graph model will take two different formats as Fig. 2 shows. In these graphs, the edges store the occurrence frequency of the events in the rule's precondition with respect to its firing frequency. For example, Fig. 2a indicates that Event 1 has occurred 44 times, the sub-sequence {Event_1 → Event_2 } 20 times whereas the rule has fired 13 times. As we will see in section 3.2.3, this information is used to infer future events.

The second step carried out by this module incrementally updates the frequency information contained in the graphs at the same time the CEP system receives and delivers events. Thus, for each new raw or derived event *enew* generated/received within the system, the module executes the procedure depicted in Algorithm 1 to update the graphs.

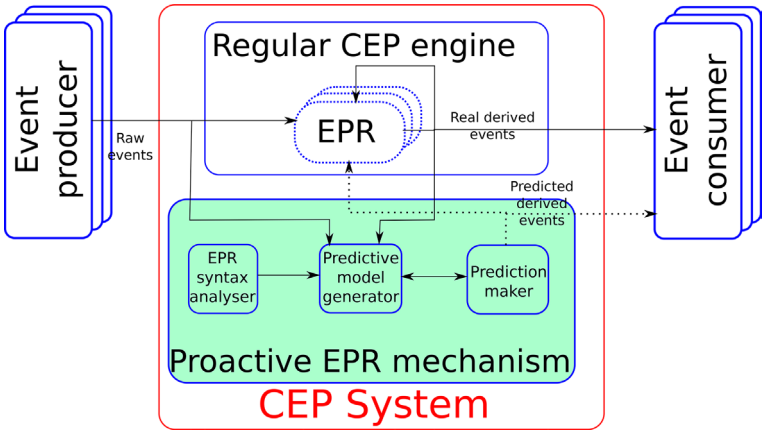


Figure 1: Integration of the solution with a regular CEP system.

In short, this procedure first searches all the graphs that contain, in some of their parts, the new event e_{new}. Then, it updates the frequency information of the edges of each selected graph (lines 1–3 of Algorithm 1). This update is performed by the function *update edges* (lines 4–14). Depending on the part of the rule where e_{new} is contained the function updates the edges in a different way. In case e_{new} is in the action part, this means the rule has has fired. Consequently, all the edges are updated by increasing their firing frequency of the rule (the right number in the edges in Fig. 2) (lines 6–9). Provided *enew* is in the condition part (lines 10–16), we first append the new event to the condition state of the rule (line 11). This state lists the events within the rule's condition that have already occurred. Once such state has been enlarged, the edge associated with the current rule's state is updated (lines 12–13). Finally, the edge's frequency solely associated with the *enew* is also incremented (lines 14–16).

For the sake of clarity, Table 1 shows how the graph in Fig. 2b would be updated in case the sequence `(Event_2, Event_1, Event_3, Derived_Event, Event_1)` is received.

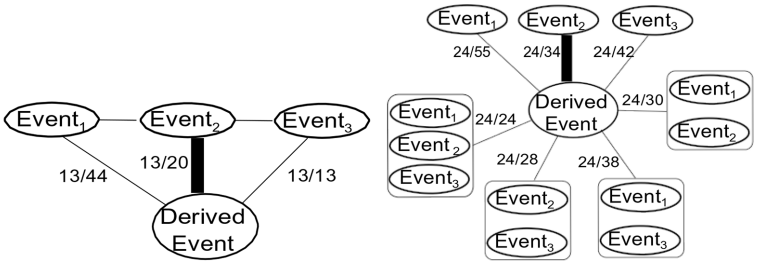
All in all, we can see that the module updates the graphs on the fly at the same time the events flow through the system. Consequently, it does not rely on any type of previously gathered historic data.

3.2.3 Prediction maker

This is the module in charge of actually predicting the events. To do so, it uses the frequency information contained in the graphs and the condition state of each rule. The process is summarized in Algorithm 2.

First of all, this procedure extracts, for each new raw or derived event e_{new}, the set of graphs G_e where the event is involved (line 1 Algorithm of 2). Then, for each graph *g*, the prediction mechanism is launched by invoking the function *make prediction* (lines 3–4). This function (lines 6–18) takes the current condition of the rule r_{cond}^{state} and uses its associated edge (*E*) to calculate the probability *p* of the derived event that will be generated by the rule according to its current state (lines 9–13). To do that, a domain-dependent parameter p_{min} defines the minimum probability to actually make a prediction.

Keeping on with the event sequence of Table 1 and provided that *pmin*is set to 0.7, the mechanism at instant $t + 1$ calculates that the proba-bility *p* of the edge associated with *rstate* is 0.77 $\left(=\frac{24}{31}\right)$. Therefore, the system anticipates the activation of the rule and generates its derived event `Derived_Event` at instant $t + 1$ before `Event_3` is delivered at instant $t + 2$ which is the one that actually causes the reaction-based activation of the rule.



(a) Graph of a *sequence-based* rule with three-event sequence in its precondition (Event_1 → Event 2 → Event 3)

(b) Graph of an *intersection-based* rule a with three events in its pre-condition (Event_1 and Event_2 and Event_3)

Figure 2: Examples of predictive graph models generated by the component. The nodes represent the events of the rule precondition and the derived one.

Algorithm 1: EPR graphs update procedure.

Input: G: set of EPR graphs, e_{new} : new event.

1 $G_e \leftarrow$ find(G; e_{new})
2 **foreach** $g \in \boldsymbol{G_e}$ **do**
3 update_edges(g,e_{new}, p)
4 **function** *update_edges*(g, e_{new})
5 $e^{g}_{type} \leftarrow g.type\_of\ (e_{new})$
6 **if** e^{g}_{type} == *action* **then**
7 $g.EPR.cond\_state \leftarrow \emptyset$
8 **foreach** $\epsilon \in g.edges$ **do**
9 ϵ.num_rule_fires++
10 **else if** e^{g}_{type} == *condition* **then**
11 $r^{state}_{cond} \leftarrow g.EPR.cond\_state \cup e_{new}$
12 $\epsilon \leftarrow g.edge\_of\ (r^{state}_{cond})$
13 ϵ.frequency++
14 $\epsilon \leftarrow g.edge\_of(e_{new})$
15 ϵ.frequency++
16 $g.EPR.cond\_state \leftarrow r^{state}_{cond}$

Table 1: Example of how the graph in Fig. 2(b) would be updated.

Time	e_{new}	r^{state}_{cond}	Updated edges	New Freq.
t	Event 2	{Event 2}	Event 2 - Derived Event	24|34→24|35
t + 1	Event 1	{Event 2, Event 1}	Event 1 - Derived Event	24|55→24|**56**
			{Event 1, Event 2} - Derived Event	24|30→24|**31**
t + 2	Event 3	{Event 2, Event 1, Event 3}	Event 3 - Derived Event	24|42→ 24|**43**
			{Event 1, Event 2, Event 3} - Derived Event	24|24→ 24|**25**
t + 3	Derived Event	∅	All	24|*→ **25**|*
t + 4	Event 1	{Event 1}	Event 1 - Derived Event	25|56→25|**57**

In the end, all the predicted events are injected into the CEP engine so that they can be processed by all the EPRs. In that sense, how such rules must deal with the probabilistic features of the predicted events is left as future work.

4 PRELIMINARY EVALUATION

4.1 Target scenario

In order to provide a preliminary test of our proposal, we have made use of the CEP system proposed in [8] to discover the occupants of a vehicle by means of some of its built-in sensors. One of the EPRs of such system intends to perceive if there is a person in the driver seat. To do that, it is activated when detects that the driver door has been opened, then the seat belt in the driver side has been fasten and finally the vehicle's engine has been turned on. The pseudocode is as follows,

```
CONDITION     OpenDriverDoorEvent ->
              FastenDriverSeatBeltEvent ->
              TurnOnVehicleEngine
ACTION        new DriverOccupantEvent();
```

This *sequence-based* EPR was handled by our mechanism so as to anticipate the occupancy in the driver seat.

Algorithm 2: Event prediction procedure.

Input: G: set of EPR graphs, e_{new} : new event, p_{min} : minimum probability.

Outpu: ε_{pred}: set of predicted derived events.

$\varepsilon_{pred} \leftarrow \emptyset$
$\boldsymbol{G_e} \leftarrow$ find (G, e_{new})
foreach $g \in \boldsymbol{G_e}$ **do**
  $\varepsilon_{pred} \leftarrow \varepsilon_{pred} \cup$ make_prediction (g, e_{new}, p_{min})
return ε_{pred}
function *make_prediction* (g, e_{new}, p_{min})
$e^{g}_{type} \leftarrow g.type\_of\,(e_{new})$
if $e^{g}_{type} == condition$ **then**
  $r^{state}_{cond} \leftarrow g{:}EPR{:}cond\_state \cup e_{new}$
  $\varepsilon \leftarrow g.edge\_of\,(r^{state}_{cond})$
  $f \leftarrow \in$. num_rule_fires
  $nrf \leftarrow \in$. num_rule_fires
  $p \leftarrow \dfrac{f}{nrf}$
  if $p \geq p_{min}$ **then**
    $e_{pred} \leftarrow g$. EPR.derived event
    $e_{pred}.prob \leftarrow p$
    return e_{pred}
  return $\emptyset$

4.2 Measurements

Two measurements have been used to evaluate our proposal,

$$precision = \frac{\#\,correct\_predictions}{\#\,correct\_predictions + \#\,incorrect\_predictions}$$

$$recall = \frac{\#\,correct\_predictions}{\#\,correct\_predictions + \#\,missing\ \ predictions}$$

where *#correct_ predictions* is the number of correctly predicted Driver Occupant Events, *#incorrect_predictions* is the number of Driver Occupant Events predicted by our mechanism that actually the rule did not generate and *#missing_predictions* refers to the number of Driver Occupant Events delivered by the rule but not predicted by our solution.

In addition to that, we also evaluated the anticipation capabilities of the mechanism by measuring the time margin between the moment at which the solution anticipates an event and the moment at which the rule actually generates the real event.

4.3 Results

Table 2 depicts these three measurements for different values of p_{min}.

As we can see, the system achieved, in all the cases a suitable level of prediction both in terms of accuracy and anticipation time. More in detail, the mechanism was able to detect the presence of the vehicle driver (by generating a predicted Driver Occupant Event) roughly two seconds in advance. This anticipation of the driver detection can be quite useful in a road-traffic management scenario to start certain transition services even more quickly.

5 CONCLUSIONS AND FUTURE WORK

CEP is nowadays a promising paradigm within the Big Data domain in order to deal with large flows of information in a timely manner. In that sense, a prominent line of work within the CEP field intends to enrich it with proactive capabilities by using external predictive analysis tools. In this scope, the present work is a first step towards a more integrated predictive solution in CEP.

Future work will focus on two courses of action. To begin with, new types of EPRs will be included by the mechanism of prediction. This would imply to consider foremost elements of CEP like sliding windows. Secondly, the current value of the events' attributes will be also considered in the production process so that the proposed mechanism will not only predict future events but also their associated details.

Table 2: Preliminary evaluation with respect the driver-detection EPR.

Parameter	$p_{min} = 0.3$	$p_{min} = 0.5$	$p_{min} = 0.8$
Precision	0.74	0.82	0.93
Recall	0.77	0.75	0.71
Anticipation time	2.2s	2.1s	2.2s

ACKNOWLEDGEMENTS

This paper has been also possible partially by the European Commission through the H2020-ENTROPY-649849 and the Spanish National Project CICYT EDISON (TIN2014-52099-R) granted by the Ministry of Economy and Competitiveness of Spain (including ERDF support).

REFERENCES

[1] IBM, Big data and analytics, available at http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html, 2015.

[2] Laney, D., 3D data management: controlling data volume, velocity, and variety. Technical report, Gartner Inc., 2010.

[3] Etzion, O. & Niblett, P., *Event Processing in Action,* Manning Publications, 2010.

[4] Wasserkrug, S., Gal, A., Etzion, O. & Turchin, Y., Complex event processing over uncertain data. *Proceedings of the Second International Conference on Distributed Event-based Systems*, ACM, pp. 253–264, 2008.
http://dx.doi.org/10.1145/1385989.1386022

[5] Fülöp, L.J., Beszédes, A., Tóth, G., Demeter, H., Vidács, L. & Farkas, L., Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics. *Proceedings of the Fifth Balkan Conference in Informatics*, ACM: New York, NY, USA, BCI 12, pp. 26–31, 2012.
http://dx.doi.org/10.1145/2371316.2371323

[6] Wang, Y. & Cao, K., A proactive complex event processing method for large-scale transportation internet of things. *International Journal of Distributed Sensor Networks*, **2014**, 2014.

[7] Nechifor, S., Trnauc, B., Sasu, L., Puiu, D., Petrescu, A., Teutsch, J., Waterfeld, W. & Moldoveanu, F., Autonomic monitoring approach based on cep and ml for logistic of sensitive goods. *IEEE 18th International Conference on Intelligent Engineering Systems INES 2014*, pp. 67–72, 2014.
http://dx.doi.org/10.1109/ines.2014.6909343

[8] Terroso-Saenz, F., Valdes-Vela, M., Campuzano, F., Botia, J.A. & Skarmeta-Gmez, A.F., A complex event processing approach to perceive the vehicular context. *Information Fusion*, **21**, pp. 187–209, 2015.
http://dx.doi.org/10.1016/j.inffus.2012.08.008