Predictive Analysis of BPM tasks with autoCEP

Raef Mousheimish[1,2], Yehia Taher[1], and Karine Zeitouni[1]

[1] DAVID Laboratory
University of Versailles
78000 Versailles, France
[2] Fondation des Sciences du Patrimoine
Labex Patrima
{firstName.lastName}@uvsq.fr

Abstract. BPM engines are designed to follow a coarse-grained processing methodology, which is not enough to manage long-running activities. To cope with this, Complex Event Processing (CEP) is exploited. However, the integration of the two fields had always been intricate. This paper demonstrates an easy tool to work out this integration.

Keywords: Business Process Management, Complex Event Processing, Real-time Analysis, Prediction

1 Introduction

Existing BPM engines [1] manage the entire process in an activity-oriented fashion. This is very advantageous to administer the flows of procedures, keep them compliant with agreements, logs, and monitor the beginning and end of tasks. However, engines give basically little or no knowledge about what is happening inside long-running tasks. Even though, most agreement violations could potentially stem from the inside of these tasks, BPM users have no possibility to predict these violations. Given the ubiquity of sensors nowadays and the availability of measurements that otherwise were unattainable before, business users are actually seeking more insight that could help to take proactive measures while long activities are executing.

To compensate for this shortcoming, many research initiatives integrate the CEP technology within BPM engines [1–3]. Even though these initiatives proposed various conceptual and practical solutions to the BPM/CEP integration, they have mainly focused on design-time integration. In other words, they annotate process models with all-purpose events at design-time. This practice does not support predictions while tasks are executing.

Up to this day, no real attempt was made to integrate **predictive** CEP capabilities inside executing long activities. We argue that the explicit reason for this is the overwhelming complexity that this integration could yield. Since in the CEP world rules are written manually, an easy-to-use solution of integrating on-th-fly such rules is not evident. And BPM tech users would find themselves in urge to master yet another complex domain with intricate jargons.

This paper demonstrates an easy-to-use tool to fulfill a BPM/CEP integration. This integration predicts in-activities violations, and therefore it helps to realize a proactive BPM. Apart from its predictive capabilities, our approach is also generic, in a sense that it works whenever classified time series data sets exist, regardless of the application domain. This data set contains the history of executions of monitorable activities, and it constitutes the training data to learn complex and predictive CEP rules.

In this work, we make use of the concept of **contextual templates** to tackle context-dependent monitorable tasks, and we demonstrate how **autoCEP** [4]- an innovative approach that automatically constructs an equipped CEP engine with predictive capabilities - could be easily integrated into BPM engines. **AutoCEP** heavily counts on time series data mining techniques, and technical details about it could be found in one of our recent publication [4].

2 Motivating Scenario

In the manufacturing of semiconductor microelectronics, several sensors to capture important measurements are deployed on the machines that etch the wafers. After the creation of one silicon wafer, it is manually checked if it is normal or abnormal, and then the sensor data related to this specific wafer are classified accordingly. Such workflow is simplified in Figure. 1. The *manufacture wafer* task is long-running and beyond the reach of the BPM engine.

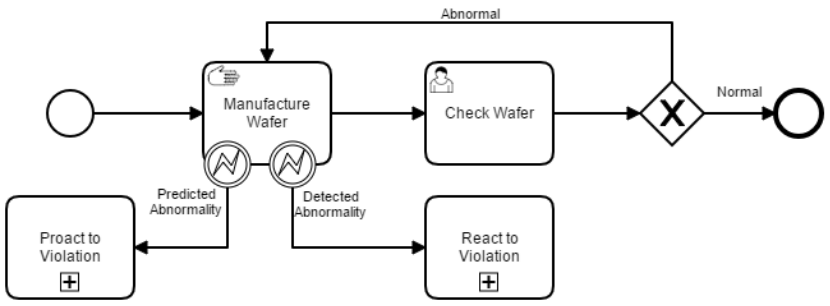


Fig. 1. Manufacturing Business Process

Currently, one could have the (un)interrupting error event that is attached to the manufacture task. Thanks to this, reactive procedures could be later executed (React to Violation sub-process in figure 1). In contrast, what could be achieved using our approach is a proactive management. The *manufacture wafer* task could have another (un)interrupting event, however this time for a predicted situation. Therefore, the management of the process could be carried out proactively (Proact to Violation sub-process in figure 1).

The strong point to emphasize is that this is not going to be achieved through an ad-hoc support, but it could be done easily in many domains and for any number of situations. The CEP engine will work automatically and signal any violation that it is trained to predict.

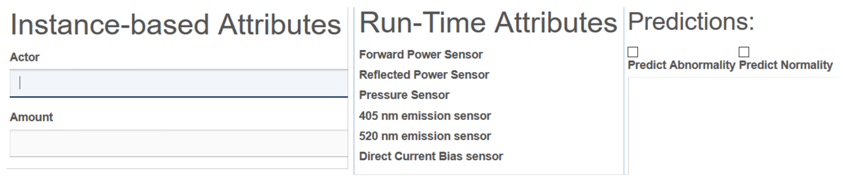
3 Contextual Templates

A main concept that paves the way for a better management of monitorable and long-running activities is contextual templates. Templates are supposed to wrap such activities in order for the CEP engine to infer what should predicted for this specific instance. This practice constitutes a way of specializing the management of processes on the instance level).

Templates comprise two types of attributes in addition to a prediction section. The first type is *instance-based attributes*. These are static data that are useful to configure the engine. The second type is *run-time attributes*. They constitute the attributes that the CEP engine should monitor in real-time.

The other part of a template is *the prediction section* where checkboxes for each situation (class) need to be provided. The user can select what situations he/she is interested in predicting for this specific instance, and later on the CEP engine will make sure to predict and signal them. Figure 2 shows an example template for manufacturing activities. It is obvious that there are six sensors deployed on the machine. The CEP engine will seek to provide real-time values from each sensor during the execution of the manufacture task. Regarding the predictions, there are two situations or classes in this case because historical wafer manufacturing scenarios are classified as normal or abnormal. Selecting some of these classes will notify the CEP engine that users are interested in predicting. Afterwards, the CEP engine will work its magic to predict these classes in an automatic way and without any manually written CEP code.

Templates are extensible, flexible, and can be easily created using current BPM engines. All typical engines provide an easy way (usually HTML forms) in order to create templates to assign values to process variables.


Instance-based Attributes

Actor

Amount

Run-Time Attributes

Forward Power Sensor

Reflected Power Sensor

Pressure Sensor

405 nm emission sensor

520 nm emission sensor

Direct Current Bias sensor

Predictions:

☐ Predict Abnormality ☐ Predict Normality

Fig. 2. Example of a Template for a Manufacturing Activity

4 Integrating autoCEP into BPM

The learning part of **autoCEP** runs over historical scenarios and produces predictive patterns. The learned patterns are saved to a local repository. Afterwards, when the process starts executing, users can check from the template the classes they are interested to predict. For instance, in the manufacturing scenario, it makes sense to predict abnormality. When the execution flow reaches the monitorable activity, e.g. manufacture, whose behavior we aim to predict, the BPM engine will trigger **autoCEP** while pointing it to the location where the temporal patterns are saved. Such a pointing could be done through the instance-based attributes of the template, or easily through the **autoCEP** API. **AutoCEP** then transforms on-the-fly the different patterns into predictive CEP rules, it configures the CEP engine to predict for the checked classes in the template, it runs the engine, and starts real-time monitoring and analysis. Subsequently, whenever one of the checked classes (in the template) is predicted, **autoCEP** will signal an event to the BPM engine (e.g., throw a BPM error with the code equals to the name of the class). Finally, attached events to the monitorable tasks (as shown in the scenario section, fig. 1) with the same code as the class will catch the error event dispatched by **autoCEP**, and the management could be carried out proactively.

BPM users are not responsible of writing CEP rules. They are rather left to focus on the design of their business processes. The only requirement is to name the process variables of the checkboxes in the prediction section of the template and the codes of the attached events, the same as the existing classes from the historical scenarios. Given this simple setup, the interaction between the BPM and the CEP engines will be done behind the scenes.

5 Demonstration

A video that presents our demonstration and puts the whole system into action (with the manufacturing scenario) could be found on our website [3]. Moreover, the website contains the source code, links to the data sets, and more experiments to assess the tool from data mining perspectives.

What we have proposed so far could be done in different existing BPM engines. In our implementation we exploited the open-source Camunda engine[4] to this end. After deploying and configuring the process (in BPMN format), executing it will trigger **autoCEP**. Then **autoCEP** will go side by side with Camunda to provide real-time analysis and prediction for the manufacturing task.

Figure 3 presents two portions of two running instances of the manufacturing process. On the left side of the figure, a normal test instance was streamed through **autoCEP**, correctly no abnormality was predicted, and the flow of the process continued normally. This is shown with the execution token inside the Camunda engine cockpit on the *No Predicted Abnormality* task. On the right side

[3] https://goo.gl/2aDHyu
[4] https://camunda.org/

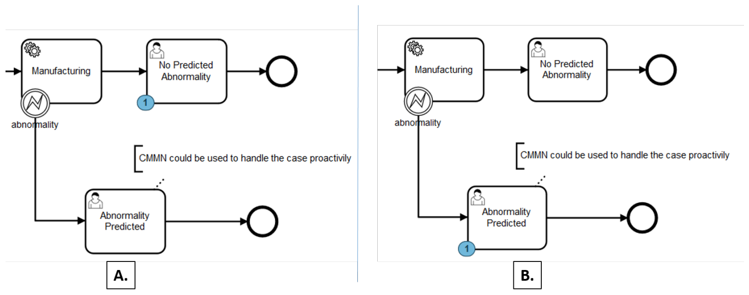


Fig. 3. A. Normal Instance B. Abnormal Instance

of figure 3, an abnormal test instance was streamed, and as shown the execution of the process was interrupted by the attached error event - the execution token on the *Abnormality Predicted* task. Thanks to this easy integration with CEP, proactive measures could now be designed and triggered.

6 Conclusion

In this paper, we presented a new technique to put **autoCEP** in service of the BPM domain in order to allow for more fine-grained management, and prediction when dealing with long-running activities. The overall tool could be considered as an easy way to integrate BPM and CEP, a problem that is lately storming the research area of BPM.

Predicting situations was discussed in this demo, however proposing adaptation countermeasures is still out of its reach. Therefore, further researches are to be continued in this direction.

References

1. A. Baumgrass, D. Ciccio, C. Claudio, R. Dijkman, M. Hewelt, J. J. Mendling, A. A. Meyer, S. S. Pourmirza, M. M. Weske, and T. Wong. Get controller and unicorn: Event-driven process execution and monitoring in logistics. CEUR Workshop Proceedings, 2015.
2. C. Cabanillas, A. Baumgrass, J. Mendling, P. Rogetzer, and B. Bellovoda. Towards the enhancement of business process monitoring for complex logistics chains. In *Business Process Management Workshops*, pages 305–317. Springer, 2013.
3. F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini. Predictive monitoring of business processes. In *Advanced Information Systems Engineering*, pages 457–472. Springer, 2014.
4. R. Mousheimish, Y. Taher, and K. Zeitouni. Automatic learning of predictive cep rules: Bridging the gap between data mining and complex event processing. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. ACM, 2017.