Predictive Task Monitoring for Business Processes*

Cristina Cabanillas[1], Claudio Di Ciccio[1], Jan Mendling[1], and Anne Baumgrass[2]

[1] Institute for Information Business at Vienna University of Economics and Business, Austria {cristina.cabanillas,claudio.di.ciccio,jan.mendling}@wu.ac.at
[2] Hasso Plattner Institute at the University of Potsdam, Germany anne.baumgrass@hpi.uni-potsdam.de

Abstract. Information sources providing real-time status of physical objects have drastically increased in recent times. So far, research in business process monitoring has mainly focused on checking the completion of tasks. However, the availability of real-time information allows for a more detailed tracking of individual business tasks. This paper describes a framework for controlling the safe execution of tasks and signalling possible misbehaviours at runtime. It outlines a real use case on smart logistics and the preliminary results of its application.

Keywords: Process Modelling, Process Monitoring, Support Vector Machines, Prediction, Event Processing

1 Introduction

Increasing availability of event data from mobile and sensor devices provides various opportunities for improving business operations. Technologies such as the Global Positioning System (GPS) or Radio-Frequency Identification (RFID) have been designed to provide for a better geographical traceability of vehicles and physical objects. The generated data can be integrated with information systems to support process monitoring, and with other knowledge repositories to support decision making. In this line, Business Process Management Systems (BPMSs) can be extended from reactive towards predictive process execution.

Although some approaches contribute to the conceptual integration of event processing and processes at design time [1, 10, 11] and alerting at run time when undesired behaviours occur [15], only a few aim to leverage the predictive capabilities associated with event processing [18]. Moreover, these are restricted to events stemming directly from the execution within the BPMS, missing misbehaviour patterns on the level of singular tasks associated with external events.

In this paper, we address this research gap by developing a technique for defining rich alert patterns associated with specific task types for predictive

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement 318275 (GET Service).

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

event monitoring in a BPMS. The feasibility of the framework is demonstrated by the help of a prototypical implementation on the basis of a real scenario on smart logistics with the task of transporting airfreight and alerting a diversion of the flight, which sets the basis to use it in different application scenarios.

The paper is structured as follows. Section 2 uses the case of airfreight transportation for identifying general challenges and requirements for a predictive system. Section 3 introduces the framework architecture. Section 4 defines an extension for task specifications in business processes, as basis for execution monitoring and prediction. Section 5 details the usage of supervised learning for realising predictive monitoring. Section 6 evaluates its feasibility. Section 7 discusses related work. Finally, Section 8 concludes the paper and envisions future research.

2 Predictive Monitoring of Continuous Tasks in Processes

Monitoring has mainly focused on identifying when tasks start and end. However, in various domains there are plenty of events recorded that can be utilised for the monitoring of a singular task, e.g., the task in charge of the shipment of goods in logistics chains. This type of tasks can be seen as *continuous* or *dynamic*, in contrast to *static* tasks such as signing a document or loading a container onto a truck. They require constant monitoring, as otherwise deviations from the expected behaviour might be detected too late, with undesirable consequences.

Let us describe part of a multimodal transport chain defined in the context of the EU-FP7 GET Service project. An aeroplane takes goods from the JFK International Airport (USA) to Amsterdam Airport Schiphol (NL), where they are transferred to a truck sent by a Logistics Service Provider (LSP) and transported to a destination in Utrecht (NL). The main goal of the LSP is to deliver the goods on time, for which the connection point in Amsterdam is especially critical. If the aeroplane has to divert and lands at a different airport (e.g., due to a thunderstorm near Amsterdam), the LSP has to cancel (or re-route) the truck that was sent to Schiphol, and in parallel reserve another vehicle to pick up the cargo at the new location. In order for these corrective actions to be effective, it is crucial that the LSP is aware of the aeroplane diversion as soon as possible, which implies constantly monitoring the task in charge of air transport.

Thus, the monitoring of dynamic tasks has implications that can be described in the form of challenges and, hence, requirements (RQ) for a predictive system:

RQ1 *Define monitoring points and expected behaviour.* The process model must be configured before enactment to introduce not only the monitoring points, but also the attributes to be considered, as well as the values desired for them.

RQ2 *Capture and process the information required for monitoring.* This information comes from different event and data sources. For instance, in the previous example, positioning information of the aeroplane can be obtained by connecting to data providers such as Flighradar24 or Flightstats. In case of road transport, it could be obtained from a GPS on-board device. In these cases, monitoring needs information external to the BPMS.

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

RQ3 *Normalise the information captured.* Relying on data sources implies that the information may arrive in different formats. Consequently, it must be first normalised in order to be jointly processed and generate valuable information.

RQ4 *Process event and data information.* Then, all the data must be processed and computed against the desired values configured in the executable model.

RQ5 *Identify and notify problems.* It is necessary to learn how to identify problems or abnormal behaviour as soon as they occur, and trigger proper alerts. Such an alerting mechanism can range from a yes/no notification indicating whether the behaviour is acceptable or not, to the detection of degrees of deviations, or root-cause analysis providing details of the problem.

RQ6 *Develop automatic support.* Support to deal with the previous challenges must be implemented and integrated into existing BPMSs.

3 Framework

Next, we describe the main components of a framework to monitor task execution and signal potential misbehaviours, addressing the aforementioned requirements. First, in a *modelling tool*, processes are modelled and tasks are annotated with predefined monitorable attributes (**RQ1**). Our approach for task annotation is described in Section 4. Second, a *process engine* calls external services to execute certain types of tasks, thus capturing the required data published by these sources (**RQ2**). This is available in engines such as Activiti. Third, a *Complex Event Processing (CEP) system* [10] is responsible for normalising events from different event sources, aggregating them to meaningful business events, and correlating them to task instances (**RQ3**). It must be capable of computing execution information against the specification set in the process model, too (**RQ4**), e.g., [3]. Fourth, a *deviation prediction system* aims at evaluating whether the task execution evolves as expected or deviates, and at informing proper participants (**RQ5**). Our approach to deal with this issue is explained in Section 5. The implementation of such a framework leads to the fulfilment of **RQ6**.

4 Definition of Monitorable tasks

In order to enable monitoring (**RQ1**), we propose to extend each task that needs to be monitored in a process model (hereinafter referred to as *monitorable tasks*) with a list of data attributes $\mathcal{T} = T_c \cup T_m \cup T_f$ divided into three groups: (i) constrained attributes T_c, for which each attribute t_c has an expected initial $\bar{t}_c^I$ and final $\bar{t}_c^F$ parameter, along with a threshold α; (ii) monitored attributes T_m, for which neither an initial nor a final parameter is meant to be provided, yet are monitored; and (iii) free attributes T_f, not monitored.

Constrained attributes are a subclass of monitored attributes, which are a specialisation of free attributes. Constrained attributes and monitored attributes can be continuously monitored if and only if they belong to numeric types, or

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

tuples of numeric types. In the latter case, the tuples must specify points that belong to an Euclidean space, e.g., longitude-latitude pairs. If the aforementioned conditions do not hold, continuous monitoring cannot be guaranteed. However, the initial and final parameters of constrained attributes can be confronted with the actual values, at the beginning and the end of the execution. We assume that the event sources (resp. external services) and their definition of event types are known. For instance, we know which values of an event indicate the position (e.g., longitude and latitude) and can use those to define rules.

5 Predictive Monitoring as a Classification Problem

Monitoring the execution of a task and checking its correct evolution corresponds to searching for possible anomalies in its behaviour. The current status of the execution is derived from the analysis of the task-related events. The gathered data are thus classified as safe or anomalous, i.e., whether they possibly lead to a successful completion or not. Such classification is based on a supervised learning model. To this extent, our approach adopts Support Vector Machines (SVMs).

SVMs [19, 7] classify an input object on the basis of its position in a numeric hyperspace. The hyperspace dimensions depend on the objects' features that the analyst specifies as relevant for the classification. A *decision hyperplane* is adopted by SVMs to separate the hyperspace into two regions, thus dividing the objects into the classes to be assigned. SVMs are supervised learning models in that they learn how to define the decision hyperplane on the basis of previous data. The objective of the SVM is therefore to determine the decision hyperplane which is capable of correctly classifying an input object. The learning phase of the SVM is conducted on the basis of labelled input, i.e., a set of input objects that were already classified in advance. SVMs build the hyperplane according to specific parameters, defining the degree of acceptance of outliers (ν) and how fitting the hyperplane has to be with respect to the pre-classified objects (γ). The learning phase is thus associated with an evaluation phase, where the SVM is trained using different combinations of such parameters (*grid search*). The best tuning is calibrated on the basis of key factors that the analyst decides.

The training phase of SVMs is usually the most expensive in terms of computational effort, whereas the run-time classification is known to be fast. This is due to the compact representation of the hyperplane by means of its so-called support vectors, which are typically sparse. In fact, we opted for SVMs because not only are they a widely used tool for classification problems explicitly addressing the anomaly detection, but they also allow for a fast classification at run-time, which is a key factor in our scenario (**RQ5**). In our approach, a different specialised classifier is adopted for each monitorable task template.

5.1 Event Dynamic Feature Extraction

Our input objects for the classification are events and the evolution of a task execution is reflected in the history of events. SVMs classify single objects in

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

a fixed-dimension space. Therefore, we adopted the following sub-sampling approach, aiming at representing the evolution of every monitored attribute as a scalar dimensionless value. In particular, this value corresponds to a normalised variation along a time interval. Each value represents one coordinate of the point in the feature hyperspace. The number of dimensions is thus fixed. Hence, the resulting point represents the dynamic change, and can be processed by the classifier, thus bridging the static analysis of single points in a feature space and the dynamicity of the task execution environment.

Let τ and τ' be two points in time, where $\tau' > \tau$, and $t_c(\tau)$ (resp. $t_m(\tau)$) the current value at time τ of a constrained attribute (resp. a monitored attribute). τ and τ' define an interval $I_\tau > 0$ during which the events are collected: $\tau' = \tau + I_\tau$. We define a new variable for monitored attribute, named *interval progress* ($\Delta\mathcal{P}^{\triangleright}_{t_m}$), as follows:

$$\Delta\mathcal{P}^{\triangleright}_{t_m} = \frac{\Delta\left(t_m(\tau), t_m(\tau')\right)}{\text{avg}\left\{t_m(\tau), \ldots, t_m(\tau')\right\}}$$

The feature representing the rate of change of the monitored attribute is computed in terms of the increment during the time interval, $\Delta\left(t_m(\tau), t_m(\tau')\right)$, scaled by the average value during the interval, $\text{avg}\left\{t_m(\tau), \ldots, t_m(\tau')\right\}$. We specify here that $\Delta(\cdot,\cdot)$ abstracts from the calculations needed to compute it. In the simplest case, it merely represents the subtraction between the passed values. Note, however, if the variables refer, e.g., to geographic coordinates, the increment has to be computed as a geodesic distance.

Constrained attributes are provided with initial and final values. The monitoring can thus be done on the basis of two more increments, with respect to *(i)* the final value, $\Delta\left(\bar{t}^F_c, t_c(\cdot)\right)$, and *(ii)* the initial value, $\Delta\left(\bar{t}^I_c, t_c(\cdot)\right)$. Therefore, we define two new variables for constrained attributes, named *progress from start* ($\Delta\mathcal{P}^{|\triangleright}_{t_c}$) and *progress to end* ($\Delta\mathcal{P}^{\triangleright|}_{t_c}$), as follows:

$$\Delta\mathcal{P}^{|\triangleright}_{t_c} = \frac{\Delta\left(\bar{t}^F_c, t_c(\tau')\right) - \Delta\left(\bar{t}^F_c, t_c(\tau)\right)}{\Delta\left(\bar{t}^I_c, \bar{t}^F_c\right)}$$

$$\Delta\mathcal{P}^{\triangleright|}_{t_c} = \frac{\Delta\left(\bar{t}^I_c, t_c(\tau')\right) - \Delta\left(\bar{t}^I_c, t_c(\tau)\right)}{\Delta\left(\bar{t}^I_c, \bar{t}^F_c\right)}$$

This reflects a perspective on the entire execution, as opposed to the interval progress, which considers an interval-focused view. The classification is hence made after events have been collected for I_τ time units. As a consequence, the anomaly refers to a single interval in time, whereas our approach aims at signalling whether the whole task is going to be disrupted. Consequently, there could be the need to wait more than one anomaly detection, before raising an alert. We indicate the number of consecutive anomaly detections as r.

The approach described so far covers not only the evolution of values for which a constraint was imposed at design time, but also for unconstrained monitorable attributes. This results in a more comprehensive observation of the evolution of task enactment.

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

5.2 Training the Classifier

Training data are gathered in our approach from a repository of event logs, in the form of stored sequences of events. Logs are pre-labelled as compliant or non-compliant according to the initial and final values for the constrained attributes: if and only if they are within the specified threshold for the activity, logs are considered as compliant.

The training of the SVM must be done not only based on its own parameters (ν and γ), but also with regards to the interval length I_τ and the number of sequential anomaly detections to accumulate before raising an alert, r. Section 6 exemplifies this joint training with a real use case.

As said, the objective of the training phase is to find the best tuning of parameters, in order to attain the best performance. In our case, the key drivers are accuracy and time-to-predict. Accuracy is assessed by Precision $\mathcal{P} = \frac{tp}{tp+fp}$, Recall $\mathcal{R} = \frac{tp}{tp+fn}$ and F-score $\mathcal{F} = 2 \cdot \frac{\mathcal{P} \cdot \mathcal{R}}{\mathcal{P}+\mathcal{R}}$ [14]. Respectively, true positives (tp) and false positives (fp) represent correct and incorrect classifications for tasks that are not respecting the constraints; true negatives (tn) and false negatives (fn) represent correct and incorrect classifications for tasks that are completing their execution according to the expected behaviour. Precision indicates the fraction of predicted anomalies that belong to the log of a misbehaving task. Recall denotes the fraction of misbehaviours that is classified as such. Finally, F-score is the harmonic mean of Precision and Recall measures. The time-to-predict (the second key driver in our approach), is computed as $I_\tau \cdot r$.

6 Evaluation

To study the effectiveness of our approach, we consider a real case study based on the monitorable task of airfreight transportation, as the one described in Section 2. In particular, we focus on alerting diversions, i.e., the signal to be raised when the aeroplane is going to land in an unplanned airport. This translates to the condition of violating the final coordinates of *aeroplane coordinates*, having the aeroplane position outside the *landing airport*.

In order to train the classifier, we collected 119 logs of events reporting flight data in the U.S. during May 2013 (98 regular flights, 21 diverted). Data were gathered from Flightstats, a data provider for air traffic information. Specifically, we automatically labelled as anomalous those traces ending in positions far from the expected destination. The remaining were compliant to the execution. The constrained attributes were *aeroplane position* (geographical coordinates), and the monitored attributes were the *aeroplane altitude* and *aeroplane speed*. Therefore, the classification was based on: *(i)* change rates in gained distance from the take-off airport (progress from start) and to the landing airport (progress to end), and *(ii)* interval-variations in *(a)* covered distance, *(b)* speed, and *(c)* altitude (interval progress) For the implementation, we adopted Esper as the CEP system and the Scikit-learn Python library's SVM as the automated classifier.

We performed a *grid search* in order to optimise the parameters described in Section 5. The ranges for parameter tuning were: *(i)* $I_\tau \in \{3, \ldots, 30\}$ min;

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

(ii) $r \in \{1, \ldots, 15\}$; *(iii)* $\nu \in \{0.01, \ldots, 0.25\}$; *(iv)* $\gamma \in \{2^{-10}, 2^{-9}, \ldots, 2^3\}$. The best combination turned out to be based on 7-minute-long intervals, with 3 consecutive anomalies considered as eligible for an alert. The best performing ν and γ parameters, in this configuration, were resp. 0.01 and 0.5. Remarkably, the most accurate tuning was also among the most rapid in terms of time to predict (21 minutes). Test data consisted of 192 logs from Flightstats differing from the training set (170 regular flights, 22 diverted). The best F-score was obtained again with the 7-minute-long-intervals configuration: 87.8%.

The advantage for the process stakeholders is reflected by the possibility to be aware of a possible process disruption ahead of time. As explained in Section 2, this leads to increased possibilities to recover the process and, possibly, to tangible savings. Therefore, we analysed *(i)* the difference in time between the planned arrival and the diversion alert raising, and *(ii)* the difference in time between the actual landing (in an unexpected location) and the alert raising. These measures assess the time gained by the LSP to reorganise the road transport, originally assigned to pick up cargo at the planned arrival airport. The response time gained for the predicted diversions, indicates that the approach is on average able to raise an alert 104 minutes before the originally scheduled landing time, and 64 minutes before the actual landing time. This is a significant gain in comparison to the case where LSPs have to wait for a notification of the diversion, which sometimes occurs up to two hours past the actual landing time.

7 Related Work

To the best of our knowledge, there is not any framework for predictive task monitoring in business processes. However, some of the requirements described in Section 2 have been (partially) addressed. Regarding **RQ1**, a set of patterns describing relations and dependencies of events in processes that have to be captured in process models to observe the overall process context have been introduced [4]. Some approaches have also focused on the representation of CEP in business processes [9, 11]. Continuous activities are typically defined in the logistics domain [6, 12, 16]. The approaches dealing with process monitoring usually aim at checking run-time compliance against rules [2, 5, 17, 20]. They capture (**RQ2**) and process (**RQ4**) events related to the process, but external sources are disregarded and, thus, **RQ3** too. Unlike in our approach, rule violations are detected and notified when they occur but predictive capabilities are missing (**RQ5**). Further results in the application of CEP for Business Activity Monitoring (BAM) [13, 8] present similar features as those related to compliance. Consequently, the existing automatic support referred by **RQ6** is partial.

8 Conclusion

In this paper, we presented a framework for monitoring the progress of task execution and predicting potential problems. To implement such a framework, an approach to configure monitorable tasks and a supervised learning model

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

to detect behavioural deviations were introduced. Tests conducted on real data showed evidence of accuracy and timeliness in the misbehaviour detection.

We aim to extend our evaluation using different task types and event information, and to investigate the automatic definition and adjustment of the thresholds for safe task execution to improve the classification and alerting mechanisms.

References

1. S. Appel, S. Frischbier, T. Freudenreich, and A. P. Buchmann. Event Stream Processing Units in Business Processes. In *BPM*. Springer, 2013.
2. A. Awad, G. Decker, and M. Weske. Efficient Compliance Checking Using BPMN-Q and Temporal Logic. In *BPM*. Springer, 2008.
3. M. Backmann, A. Baumgrass, N. Herzberg, A. Meyer, and M. Weske. Model-Driven Event Query Generation for Business Process Monitoring. In *ICSOC 2013 Workshops*. Springer, 2014.
4. A. Barros, G. Decker, and A. Grosskopf. Complex Events in Business Processes. In *BIS*. Springer, 2007.
5. A. Birukou, V. D'Andrea, and F. Leymann. An Integrated Solution for Runtime Compliance Governance in SOA. In *ICSOC*. Springer, 2010.
6. C. Cabanillas, A. Baumgrass, J. Mendling, P. Rogetzer, and B. Bellovoda. Towards the Enhancement of Business Process Monitoring for Complex Logistics Chains. In *BPM 2013 Workshops (PALS)*. Springer, 2013.
7. C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
8. A. Dahanayake, R. Welke, and G. Cavalheiro. Improving the Understanding of BAM Technology for Real-Time Decision Support. *IJBIS*, 7(1), December 2011.
9. G. Decker, A. Grosskopf, and A. Barros. A Graphical Notation for Modeling Complex Events in Business Processes. In *EDOC*. IEEE, 2007.
10. N. Herzberg, A. Meyer, and M. Weske. An Event Processing Platform for Business Process Management. In *EDOC*. IEEE, 2013.
11. S. Kunz, T. Fickinger, J. Prescher, and K. Spengler. Managing Complex Event Processes with Business Process Modeling Notation. In *BPMN*. Springer, 2010.
12. F. Liao, J. L. Wang, and G.-H. Yang. Reliable Robust Flight Tracking Control: an LMI Approach. *IEEE Trans. Control Systems Technology*, 10(1):76–89, 2002.
13. D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley, 2001.
14. T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
15. M. Montali, F. M. Maggi, F. Chesani, P. Mello, and W. M. P. van der Aalst. Monitoring Business Constraints with the Event Calculus. *ACM TIST*, 5(1), 2013.
16. L. X. Pang, S. Chawla, W. Liu, and Y. Zheng. On Detection of Emerging Anomalous Traffic Patterns Using GPS Data. *Data & Knowledge Engineering*, 2013.
17. R. Thullner, S. Rozsnyai, J. Schiefer, H. Obweger, and M. Suntinger. Proactive Business Process Compliance Monitoring with Event-Based Systems. In *EDOC Workshops*. IEEE, 2011.
18. W. M. P. van der Aalst, M. H. Schonenberg, and M. Song. Time Prediction Based on Process Mining. *Inf. Syst.*, 36(2), 2011.
19. V. Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer, 1982.
20. M. Weidlich, H. Ziekow, J. Mendling, O. Günther, M. Weske, and N. Desai. Event-Based Monitoring of Process Execution Violations. In *BPM*. Springer, 2011.

Pre-print copy of the manuscript published by Springer (available at link.springer.com) identified by DOI: 10.1007/978-3-319-10172-9_31

References

Cabanillas, Cristina, Claudio Di Ciccio, Jan Mendling, and Anne Baumgrass (2014). "Predictive Task Monitoring for Business Processes". In: *BPM*. Ed. by Shazia Wasim Sadiq, Pnina Soffer, and Hagen Völzer. Vol. 8659. Lecture Notes in Computer Science. Springer, pp. 424–432. ISBN: 978-3-319-10171-2. DOI: 10.1007/978-3-319-10172-9_31.

BibTeX

```
@InProceedings{   Cabanillas.etal/BPM2014:PredictiveTaskMonitoring,
  author        = {Cabanillas, Cristina and Di Ciccio, Claudio and Mendling,
                   Jan and Baumgrass, Anne},
  title         = {Predictive Task Monitoring for Business Processes},
  booktitle     = {BPM},
  year          = {2014},
  pages         = {424--432},
  publisher     = {Springer},
  crossref      = {BPM2014},
  doi           = {10.1007/978-3-319-10172-9_31},
  keywords      = {Process Modelling; Process Monitoring; Support Vector
                   Machines; Prediction; Event Processing}
}
@Proceedings{     BPM2014,
  title         = {Business Process Management - 12th International
                   Conference, {BPM} 2014, Haifa, Israel, September 7-11,
                   2014},
  year          = {2014},
  editor        = {Shazia Wasim Sadiq and Pnina Soffer and Hagen
                   V{\"{o}}lzer},
  volume        = {8659},
  series        = {Lecture Notes in Computer Science},
  publisher     = {Springer},
  isbn          = {978-3-319-10171-2},
  doi           = {10.1007/978-3-319-10172-9}
}
```