2016 IEEE 40th Annual Computer Software and Applications Conference

Toward the Support of Challenging Service Level Agreements (SLAs) in Manual and Context-Dependent Activities

Raef Mousheimish
Université de Versailles
Saint-Quentin-en-Yvelines
78000 Versailles, France
Fondation des Sciences du Patrimoine
LabEx PATRIMA
raef.mousheimish@uvsq.fr

Yehia Taher and Karine Zeitouni
Université de Versailles
Saint-Quentin-en-Yvelines
78000 Versailles, France
{*name.lastname@uvsq.fr*}

***Abstract*—Recent research initiatives in the domain of business process management such as process intelligence, monitoring, and mining have shown significant results in automated process environments. However, such techniques fall short to provide efficient solutions and support for non-fully automated business processes i.e., processes that embody dynamic, continuous, and manual activities such as in logistics. More precisely, things turn to be very challenging when it comes to the monitoring and Service Level Agreement (SLA) violation prediction throughout context-dependent and manual processes. Unlike current approaches that mainly focus on the model of the process as a whole, we shift in this work toward instance-based and specific processing for each activity depending on its context. We showcase a contextualized template-driven framework while providing the missing link of continuous monitoring and early prediction within manual activities.**

1. Introduction

Due to the popularity of service-based practices and business process management nowadays, many enterprises have shifted toward them in order to flexibly adapt on the market changes and the increased competitiveness. Progresses have been achieved and advanced techniques have seen the light, like process intelligence, monitoring, prediction, and mining [9]. These techniques are efficiently exploited in **automated execution environments** where software engines fully orchestrate the fulfillment of services. However in **manual execution environments** things tend to be more complicated and the support of advanced techniques has not yet flourished.

On the other hand, in line with the emerging concept of IoT *(Internet of Things)* and the increasing availability of event data coming from sensors and sources deployed everywhere, manual business operations could be improved. The available real-time events are seen as digital representations of the real physical world, and their integration into information systems will support SLA monitoring, prediction, and other forms of process intelligence and decision making.

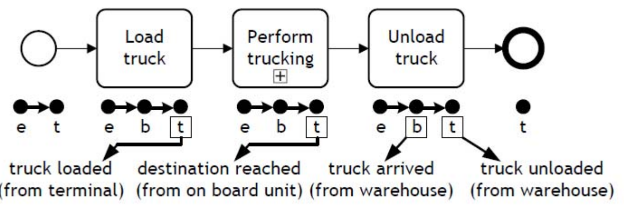
Load truck
Perform trucking
Unload truck
e
t
b
truck loaded (from terminal)
destination reached (from on board unit)
truck arrived (from warehouse)
truck unloaded (from warehouse)

Figure 1. An example of model-based monitoring points from [1]

Therefore, we recently witnessed some approaches [1], [3] that contribute to the incorporation of events within manual business processes like logistics [1], [4]. However, they all assign what we call *model-based monitoring points* (figure 1 shows an example). They annotate the process model at design-time with some monitoring points to track its progress, detect coarse-grained deviations, and sometimes predict end-to-end violations. We argue that just relying on this design-time methodology will yield some problems from context-aware perspectives. More specifically, current approaches monitor, support, and execute each instance that is stemming from the same model in an identical pattern, even though their contexts are completely different and dynamically evolving.

In this paper, we address the gap between manual processes and information systems by discussing our techniques to apply process monitoring and SLA predictions in manual environments. We propose a flexible framework that entails different contributions in the domain. Particularly: (1) We introduce the concept of contextualized templates to capture the context of each activity in the process. (2) We widen the scope of processed events to include information from external (IoT) sources, so we discuss our dynamic enrichment methodology that is driven by the template itself. (3) We take advantage of the predictive capabilities of Complex Event Processing (CEP) to anticipate SLA violations at runtime and not just signal them when they happen. Then we show some of the advancements and satisfactory results that we achieved till this point concerning our prediction mechanism.

In the remainder, a real life situation that motivates our research is first presented. Then section 3 explains the concept of templates which are the main building block of the approach. Afterward, our framework for finer management of activities is discussed in section 4. In section 5, we emphasize on our novel prediction methodology, and section 6 shows some initial experiences. Finally related work is discussed and the paper is concluded in the remaining sections.

2. Motivating Scenario

The framework could be used in different domains where challenging constraints apply. In this paper, a trucking activity in an artwork transportation process will be taken as a running example (i.e., transportation of critical goods). This kind of logistics processes is highly challenging due to the sensitive nature of the transported piece of arts. Typically, the involved parties are interested in, and they agree to constrain different factors like temperature and vibration exposures. In this field, it is important to respect these constraints, maintain a high Quality of Service (QoS), and avoid violations, or else cultural heritage will be endangered. However in real life scenarios (that we have got from our partner[1]), breaches have been registered in more than 80% of the cases.
Current management systems help to model and monitor the progress of the whole process. They treat continuous and dynamic activities like trucking with no major differences than instantaneous activities like signing a document. Even adding monitoring points at design-time will not compensate the lack of advanced capabilities, such as predicting SLA violations as early as possible.

3. Contextualized Templates

In order to specialize each instance of the process, we propose to wrap continuous and dynamic activities with contextualized templates. This practice constitutes the main enabler to achieve instance-based and context-aware activity management, as templates are flexible, they facilitate the capture of contexts, and they allow for a good run-time tracking and representation of manual activities within information systems. The template carries three types of attributes, and figure 2 illustrates an example that could be applied for our trucking activity:
Instance-Based Attributes: The attributes that fall under this category need to be filled at design-time (by the user when needed) and before executing the process. Typically they are static during the execution, and with different values for each instance of the activity.
Run-Time Attributes: These attributes are general, and they are filled with real-time values by tracking the process

1. C2RMF: A research center in Paris that is concerned with the preservation of artworks for the museums of France (http://c2rmf.fr)

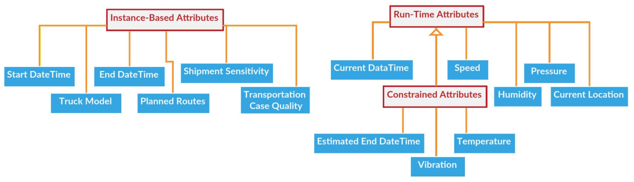

Figure 2. An Example Template for a Trucking Activity

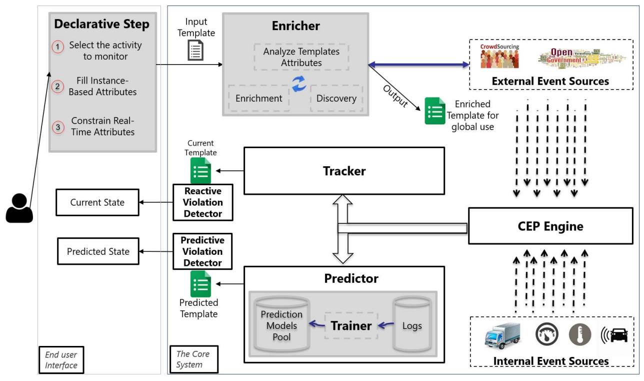

Figure 3. The Overall Architecture

at run-time.
Constrained Attributes: This group contains the Run-Time Attributes, which values may reflect signs of violations. They need to be constrained by the user. In other words, they may constitute SLA clauses, and the framework needs to continuously predict and estimate the values of these attributes in order to give insight about violations if any.

4. Context-Aware Management Framework

The goal of the framework proposed in this paper is to help the end user to get insights and predictions about critical SLA clauses. These predictions are the cornerstone to later employ proactive and preventive measures. It is necessary to note that the focus in this work is on the prediction part and not the proactive adaptation.

The overall proposed architecture is sketched in figure 3. We are going to clarify its logical workflow by separating its components into two classes: design-time (filled with light gray on the figure), and run-time.

4.1. Design-Time

4.1.1. Declarative Step. It is important to note that current business process management systems allow the creation of models that look like the one in figure 4, with no further support on how to predict SLA violations inside a manual and continuous activity (e.g., Trucking activity). Whereas the declarative steps proposed in this section will help to overcome this limitation, and to allow for a more contextualized processing for each instance of the activity.

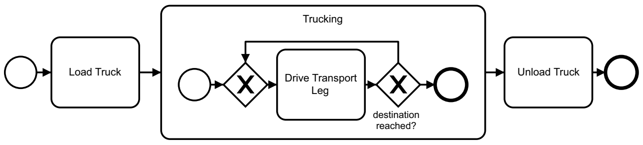

Figure 4. An Example of a Transport Business Process

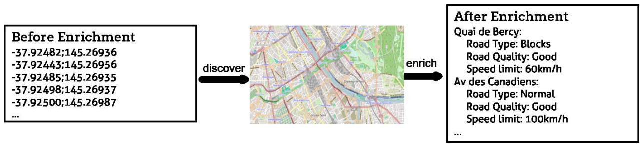

Figure 5. The Planned Routes before and after the enrichment

First the end user is required to select the continuous activity to manage on a finer level, and secondly she will be prompted with the proper template to fill its **instance-based attributes**. Finally, the user is asked to constrain the **run-time attributes** that may incarnate SLA clauses to respect. For instance, she may allocate minimum and maximum limits when applicable, e.g., $14°\text{C} \leq temperature \leq 25°\text{C}$; $vibration \leq 4g$; estimated completion time $\leq$ 23 Sept 2016 04:00 AM...(refer to section 3 for more details on template attributes and their types)

4.1.2. Enricher. In order to be aware of the context, one can not just rely on internal and known event sources (i.e., events describing the process itself, like the location of the truck...), but it is necessary to widen the scope of processed information to include external events (i.e., events describing the environment where the process is being executed, like road status...). To this end, the **Enricher** is realized. As shown in figure 3, this component processes the input template, discovers external sources, and uses them to enrich the attributes, to finally output a modified, meaningful, and enriched version of the template.

It is essential to know that our enrichment mechanism is dynamic and driven by the **constrained attributes** that are selected by the user. More precisely, to predict the values of the chosen **constrained attributes** (SLAs), specific prediction models and algorithms are employed and these models require different inputs. We split these input variables into two types, *known inputs* that are available through the template without enrichment (e.g., temperature, GPS, transport case quality, etc.), and *unknown inputs* that could not be recognized unless we employ an enrichment procedure (e.g., road qualities, permitted road speeds, etc.). As an example and to clarify our strategy, the regression function that is used to simulate vibration values over the roads (more on this in the next section) requires four parameters. They are the Road surface Quality (RQ), Road surface Type (RT), Transport Case Quality (TCQ), and the Permitted road Speed (PS), three of them are *unknown inputs* ($RQ; RT; PS$). In this case, our system triggers the need for enrichment, and the discovery process begins.
The actual enrichment logic differs depending on the required *unknown inputs*, but in general the framework uses raw data that it has in order to discover relevant external services and finally enrich the template. By reusing our regression function example, we will continue to elaborate how the enrichment is done. First and starting from the planned routes that exist in the **instance-based attributes** –typically these are a set of coordinates that specify the transport legs of the truck (raw GPS data)– the framework employs the Nominatim reverse geocoding API[2] and queries open data offered by Open Street Map[3] to shift from low-level data into higher knowledge that fit for the prediction model (the regression function in this example) as shown in figure 5.

The novel point to notice here is that this is done dynamically and differently for each instance, i.e., it depends on the environment (context) of the process and the choice of the user for **constrained attributes**.

4.1.3. Trainer. In the Predictor module, the training phase is a sub-part that is dedicated for design-time (fig. 3). At this stage and using different data mining techniques, various prediction models from the available history *logs* are trained and stored in the *prediction models pool*. It is important to note that each SLA clause (attributes that are constrained by the user) needs to be covered by a prediction model.
Currently several prediction models for some of the attributes are supported. First, temperature and humidity values are predicted using early classification on time series techniques [16], [17]. Second, future vibration values are estimated by utilizing a trained regression function that takes into account the roads and the speed of the truck. Finally, the completion time of the activity is estimated by using a delay propagation algorithm [5]. We will demonstrate the first prediction technique later in this paper (in section 5), whereas we will leave the details of the remaining two due to space limits.

4.2. Run-Time

4.2.1. CEP Engine. The **Complex Event Processing (CEP) Engine**, Esper[4] in our case, is in charge of subscribing to services, correlating, integrating, and handling the various streams of events. Its mission is straightforward, i.e., matching the incoming events against the different CEP rules that are enrolled in, and triggering the corresponding listeners.

4.2.2. Tracker & Predictor. The aforementioned CEP queries are statements registered by these two components. Therefore, after a query matches in the **CEP Engine**, the corresponding listener in the **Tracker** or the **Predictor** will be invoked.

2. http://nominatim.org. It transforms from raw GPS to meaningful names
3. http://openstreetmap.org
4. http://espertech.com/esper

The name of the **Tracker** is self-explanatory. It gets real-time readings from the engine, in order to fill the **run-time attributes** with accurate data and provide the *current template* to the user as the *current state* of the activity (fig. 3).
The **Predictor** functionality are divided into two parts. We already covered the **Trainer** at design-time. At run-time the main responsibility of the **Predictor** is to ensure the availability of all inputs that are required by the employed prediction models. These inputs could be gained from the already available and *enriched template*, or from the **CEP engine** when necessary. As an example, for the regression function that predicts vibration, inputs are available through the *enriched template*, but on the other hand the delay propagation algorithm requires real-time information like road status, accident reports, etc. that the **Predictor** can only get through the **CEP Engine**. By feeding the acquired inputs to the appropriate prediction models, the **Predictor** outputs a *predicted template* that contains predictions for all **constrained attributes**, to finally render it as the *predicted state* for the end user (fig. 3).

4.2.3. Violation Detectors. After emitting the *current and the predicted templates*, a post-processing phase is realized to check for constraint violations, where the outputted values are compared against the constraints specified by the user at design-time. Two types of detectors should be distinguished: Reactive and Predictive. The **Reactive Violation Detector** is based on the *current template* and detects in real-time the different violations that cannot be predicted or avoided (e.g., incidents or accidents). Whereas, the **Predictive Violation Detector** exploits the *predicted template* to foresee violations and risks as early as possible.

It is important now to look at the framework from a global viewpoint, and note how the usage of contextualized templates allowed the information system of the client to track the state of the application at real-time, and predict its challenging SLA deviations during execution as well.

5. CEP Rules for Early Prediction

We discuss in this section one of the data mining techniques that we have mentioned earlier, i.e., early classification on time series. To this end, we devised a novel mechanism to automatically generate CEP rules by relying on this technique. The generated prediction models could be used to predict different situations as early as possible depending on data records collected over time. For our trucking use case, it fits to anticipate atmospheric conditions, such as temperature (fig. 6), humidity, and pressure.

5.1. Early Classification over time series

5.1.1. Definitions. We denote a time series $T = \{t_1, t_2, ..., t_L\}$ of length L, $len(T) = L$, as a sequence of real values sampled at L time stamps. Each time series is associated with a class c, we write $c = Class(T)$. A subsequence s of T $(s \subset T)$, $s = \{t_i, t_{i+1}, ..., t_{i+l-1}\}$, is a sequence of continuous values from T of length $l < L$. Given two subsequences s and h of the same length l, the distance between them $dist(s, h)$ is calculated as the Euclidean distance. The distance between a time series T, $len(T) = L$ and a subsequence s, $len(s) = l$ $(l < L)$, is the minimum distance between s and all subsequences h_i, $len(h_i) = l$ of T, so it is computed as:

$$dist(s, T) = \min_{\forall i \in \{1,2,...,L-l+1\}} dist(s, h_i) \tag{1}$$

A shapelet f is a temporal pattern that should be extracted from the historical time series at design-time (training time), and that could lead the prediction procedure at run-time. More formally, $f = (s, \delta, c_f)$ where s $(len(s) = l)$ is the subsequence of real values that constitutes f, and c_f is the class of the shapelet. So at run-time when temporal events are monitored, and a subsequence h that is similar to the shapelet f is detected (i.e., similar to s that constitutes f), then the engine can predict that $Class(h) = c_f$ without waiting to obtain the whole time series after h, hence the term **early** classification. A subsequence h is considered similar to f if $dist(s, h) \leq \delta$ (δ is called the distance threshold of f).

5.1.2. Shapelet Learning Algorithm. We have implemented a java tool, where experts could specify the minimum thresholds for the attributes that they want to monitor (e.g., temperature). The software then runs over the training data set (historical time series from real scenarios), and classify them as violated (e.g., when the temperature trespasses the threshold, fig. 6) or normal (i.e., not violated). Afterward, our learning algorithm will take these classified time series as inputs, and it will output a list of highly useful shapelets for each class (violated & normal). These shapelets are subsequences of minimum lengths that are sufficient to predict if a series will continue to cause a violation or not. We will not dive in the details of our learning algorithm, instead we will just hint that we used an information gain method to learn the distance threshold δ of each shapelet, and we employed a utility score formula that takes frequency, discrimination and earliness into account in order to select the highly useful shapelets for each class. For detailed explanations, interested readers may refer to [16], [17].

5.2. CEP Rules Extraction Algorithm

After learning the most useful shapelets, another java tool[5] will take them as inputs, algorithmically transform them into CEP rules, and finally register them in the Esper CEP engine in an automatic manner so they will be ready for run-time usage. In general, the CEP rules extraction algorithm creates a CEP rule for each shapelet, it infers the time window of the rule from the concerned shapelet length, and then it attaches similarity measure calculators

5. https://github.com/rmgitting/autoCEP. all the source code with a demonstrative video

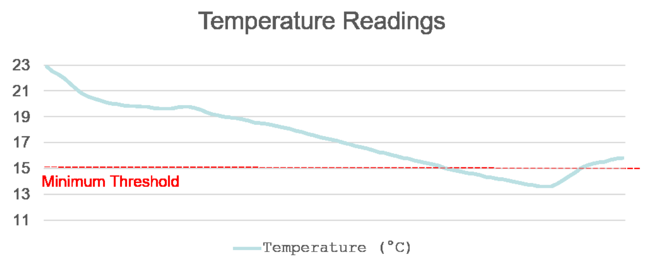

Figure 6. Temperature Distribution over Time

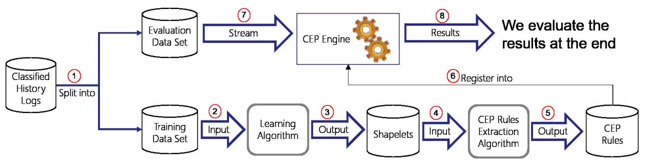

Figure 7. The Algorithms and the Evaluation Mechanism

(Euclidean distance in our case) to test if the sensed events within the time window are similar to the shapelet or not. In case a similarity is detected, then the engine will alert about a possible future violation.

6. Evaluation

The prediction algorithm is tested on real life scenarios from the domain of artworks transportation. Figure 7 summarizes the evaluation mechanism that we have followed. In general, the available data are divided into training and evaluation data sets, the former is used to learn the patterns, the latter is streamed into the engine to test, and finally the results are evaluated.

6.1. Results

Two of the important factors that we are really interested in are the exactitude and the earliness of the predictions. To calculate the $Avg.f\text{-}score$ (exactitude) we employed this formula (where C designates the set of classes, i.e., violated and normal): $\frac{1}{|C|}\sum_{cl \in C} \frac{2 \times precision(cl) \times recall(cl)}{precision(cl)+recall(cl)}$, with $precision(cl) = \frac{TP}{TP+FP}$ and $recall(cl) = \frac{TP}{TP+FN}$.
On the other hand, the earliness is computed from the average percentage of time points needed to make the predictions. Giving a testing dataset D, a timeseries T, and the Earliest Matching Time of T as $EMT(T)$ (this factor represents the time points when the prediction is done), we calculate the earliness percentage as: $\frac{1}{|D|}\sum_{T \in D} \frac{EMT(T)}{len(T)}$.

Table 1 describes our training and evaluation data sets. The numbers inside the cells of table 2 indicates the number of instances that belongs to each category (column), when we tested them against the rules that predict the violated class only.

Furthermore, table 3 shows the average f-score, and the earliness (as a percentage) of the approach.

TABLE 1. Test and Evaluation Data Sets

	Violated Scenarios	Normal Scenarios	Longest Series	Shortest Series
Training	16	17	451	51
Evaluation	17	17	460	39

TABLE 2. Earliness & Exactitude: Number of Instances

Positives		Negatives		Earliness (in minutes)			
True (TP)	False (FP)	True (TN)	False (FN)	≥ 60	≥ 30	≥ 5	late
17	1	16	0	9	5	1	2

7. Related Works

The increasing availability of cheap and pervasive sensor technology, in addition to the seamless broadband connectivity almost everywhere, have promoted the expectations for the Internet of Things (IoT). In line with this prominent concept, many recent proposals have targeted the improvement of manual business process management. More precisely, enhanced monitoring, fine-grained violations prediction, and proactive adaptations are storming the domain.
Monitoring has always been the cornerstone to achieve more advanced support like prediction and adaptation [8], and researchers have agreed that efficiently monitoring manual processes and their contexts is critical yet it is beyond the reach of current business process management systems [11], [12]. Therefore, they attempted to extend the existing techniques, by integrating complex event processing capabilities while managing their processes. However the main shortcomings lie in the model-based processing methodology that they follow [1], [3], [4], [10], [11], [15]. By just focusing on the model, the aforementioned approaches assign monitoring points over the process model, so that the monitoring, subsequently the prediction and the adaptation, are executed upon reaching these points. For instance, some approaches (like [3], [4], [11]) offer the possibility to explicitly annotate the process model with monitoring points at design-time, whereas others (like [6], [7], [10]) allocate these points implicitly, e.g., before or/and after each task in the process. This methodology brings in some serious limitations. Firstly, instances of the same model are managed identically, because for each one of them the same universal events are monitored, even thought in dynamic environments contexts are different, and each executing process should be handled

TABLE 3. Earliness (%) & Exactitude (f-score)

Average f-score	Earliness (%)
0.97	48

accordingly. Secondly, because of the generalized processing that we just mentioned, current approaches (like [13], [14]) usually disregard contextual events coming from external sources. Thirdly, in-activities violations can not be detected or predicted unless we reach the next monitoring point, so even predictive approaches like [3], [10] are still reactive in some ways. Finally, predicting end-to-end violations depending on monitoring points may lead to time issues (i.e., remaining time to adapt), consequently it may produce late and ineffective predictions. The last two points highlight the lack of real predictive capabilities within existing approaches.

On the other hand, The Green European Transportation Service or the GET service[6], is one of the most recent European projects that is held in the domain of logistics. It incorporates efforts to enhance manual business process management in order to efficiently support transportation missions. Despite the large scope of problems and publications that GET embraces, this approach [2] is of utmost importance. To the best of our knowledge, it is the only approach that inspects almost the same question that we are trying to address in our work. Authors distinguished monitorable tasks, along with their attributes, which need to be continuously monitored in a logistics process. Then, during execution, they monitor these attributes in a interval-based scheme, and by counting on a trained support vector machine classifier, they detect violations if any. Although this approach is related, yet it was specific to one use case (i.e., detecting the divergence of a flight as early as possible). On the contrary, in this work we elaborate with details the problems of current approaches and the limitations that they impose, we propose a flexible template-driven framework that could be configured and utilized to tackle many dynamic and continuous activities, and we completely predict challenging constraint violations.

8. Conclusion

We introduced a novel framework that could address the gap between manual processes and information systems. Its main goal is to extend current model-based methodologies, in order to reach a more finer level of processing, and specialize the handling of each case according to its context. We proposed the usage of contextual templates that go beyond design-time monitoring points to exercise more advanced support for continuous and dynamic activities. We showed how our system progresses from current works that detect coarse-grained violations and predict end-to-end violations, to completely predict in-activities violations before they happen.

Furthermore, we introduced an architecture that implements our framework, we detailed its workflow along with its components, and we demonstrated the functionality of each one of them. In addition, we proved the advantages of the prediction algorithm by evaluating it against real-life scenarios and showing some of the satisfactory results that it yielded. Our proposed architecture could be considered as a cornerstone to later employ more advanced techniques such as risk assessment, context-aware recommendation, and proactive adaptation.

Fully implementing the framework, extending to support more continuous activities, and appending a component for proactive adaptations and recommendations, are all plans to realize on our future schedule.

6. http://getservice-project.eu/

References

[1] N. Herzberg, A. Meyer, and M. Weske. An event processing platform for business process management. EDOC, IEEE, 107-116, 2013.

[2] C. Cabanillas, C. Di Ciccio, J. Mendling, and A. Baumgrass. Predictive Task Monitoring for Business Processes. BPM. Springer, 424-432, 2014.

[3] P. Leitner, J. Ferner, W. Hummer, and S. Dustdar. Data-driven and automated prediction of service level agreement violations in service compositions. Distributed and Parallel Databases, 447-470, 2013.

[4] T. Metzke, A. Rogge-Solti, A. Baumgrass, J. Mendling, and M. Weske. Enabling Semantic Complex Event Processing in the Domain of Logistics. ICSOC Workshops. Springer, 419-431, 2013.

[5] R. Mousheimish, Y. Taher , and B. Finance. Towards smart logistics processes: a predictive monitoring and proactive adaptation approach. In Proceedings of the 2015 International Conference on Software and System Process 2015 Aug 24. ACM.

[6] M. Andrea, A. Russo, and M. Mecella. Planlets: Automatically Recovering Dynamic Processes in YAWL. OTM 2012. Springer Berlin, 2012. pp. 268–286.

[7] C. Wang, and J.-L. Pazat. A two-phase Online Prediction Approach for Accurate and Timely Adaptation Decision. IEEE Ninth International Conference on Service Computing (SCC). pp. 218–225. 2012.

[8] J. Kephart, and D. Chess. The vision of autonomic computing. Computer, vol. 26, no. 1, pp. 41–50, Jan 2003.

[9] W. M. P. van der Aalst, et al. Process mining manifesto, BPM Workshops. Springer, 169-194, 2012.

[10] W. M. P. van der Aalst, M. Pesic, and M. Song. Beyond process mining: from the past to present and future. In Advanced Information Systems Engineering. Springer, 38-52, 2010.

[11] A. Baumgrass, C. Di Ciccio, R. Dijkman, M. Hewelt, J. Mendling, A. Meyer, S. Pourmirza, M. Weske, and T. Y. Wong. GET Controller and UNICORN: Event-driven Process Execution and Monitoring in Logistics. 2015.

[12] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. Fundamentals of Business Process Management. Springer, 2013.

[13] 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.

[14] M. Weidlich, H. Ziekow, J. Mendling, O. Günther, M. Weske, and N. Desai. Event-Based Monitoring of Process Execution Violations. In BPM. Springer, 2011.

[15] N. Herzberg, and A. Meyer. Improving Process Monitoring and Progress Prediction with Data State Transition Events. in ZEUS, pp. 20–23. 2013.

[16] Y.-F. Lin, H.-H. Chen, V. S Tseng, and J. Pei. Reliable Early Classification on Multivariate Time Series with Numerical and Categorical Attributes. Advances in Knowledge Discovery and Data Mining. pp. 199–211. Springer, 2015.

[17] M. F. Ghalwash, and Z. Obradovic. Early Classification of Multivariate Temporal Observations by Extraction of Interpretable Shapelets. BMC Bioinformatics, vol. 13, n. 1. 2012.