A ProM Operational Support Provider for Predictive Monitoring of Business Processes

Marco Federici[1,2], Williams Rizzi[1,2], Chiara Di Francescomarino[1], Marlon Dumas[3], Chiara Ghidini[1], Fabrizio Maria Maggi[3], and Irene Teinemaa[3]

[1] FBK-IRST, Italy.
{federici,wrizzi,dfmchiara,ghidini}@fbk.eu
[2] University of Trento, Italy.
[3] University of Tartu, Estonia.
{marlon.dumas,f.m.maggi,irheta}@ut.ee

Abstract. Predictive process monitoring is concerned with exploiting event logs to predict how running (uncompleted) cases will unfold up to their completion. In this paper, we propose an implementation in the ProM toolset of a predictive process monitoring framework for estimating the probability that an ongoing case will lead to a certain outcome among a set of possible outcomes. An outcome refers to a label associated to completed cases, like, for example, a label indicating that a given case completed "on time" (with respect to a given desired duration) or "late", or a label indicating that a given case led to a customer complaint or not. The framework takes into account both the sequence of events observed in the current trace, as well as data attributes associated to these events. The prediction problem is approached in two phases. First, prefixes of previous traces are clustered according to control flow information. Secondly, a classifier is built for each cluster to discriminate among a set of possible outcomes. At runtime, a prediction is made on a running case by mapping it to a cluster and applying the corresponding classifier.

1 Introduction

Often, questions and predictive challenges can arise during the execution of business processes. For example, in a medical process execution a doctor may ponder whether a surgery, a pharmacological therapy or a manipulation is the best choice to be made in order to guarantee the patient recovery. *Predictive business process monitoring* [6] is a family of techniques that apply what we do in everyday life to the field of business processes. In particular, predictive process monitoring exploits event logs, which are more and more widespread in modern information systems, to predict how current (uncompleted) cases will unfold up to their completion. A predictive process monitor allows users to predict the most likely outcome of the ongoing case. In this context, an outcome could be, for example, the timely completion of the case with respect to a deadline (versus late completion), or the fulfillment of a desired business goal (e.g., a sales process

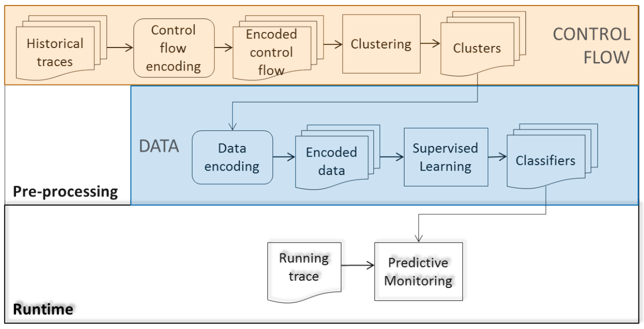


Fig. 1: Predictive process monitoring framework

leading to an order, or an issue handling process leading to successful resolution). Based on the analysis of execution traces, the monitor provides the user with estimations of the likelihood of achieving a given outcome for a running case.

In this paper, we describe an implementation in the ProM process mining toolset of a general customizable predictive process monitoring framework [3] that allows users to assign a "label" (outcome) to an ongoing case based on: (i) a prefix thereof; and (ii) a set of labeled completed sequences (the "history"). ProM provides a generic Operational Support (OS) environment [2,7] that allows the tool to interact with external workflow management systems at runtime. A stream of events coming from a workflow management system is received by an OS service. The OS service is connected to a set of OS providers implementing different types of analysis that can be performed online on the stream. The predictive process monitoring framework has been implemented as an OS provider.

2 Framework and Tool

The framework requires as input a set of past executions of the process. Based on the information extracted from such execution traces (sequences of events with their associated payload, i.e., attribute-value pairs), it tries to predict how currently evolving executions will develop in the future. To this aim, before the process execution, an automated pre-processing phase is carried out. In such a phase, state-of-the-art approaches for clustering and classification are applied to the historical data in order to (i) identify and group historical trace prefixes with a similar control flow, i.e., to delimitate the search space on the control flow base (clustering from a control flow perspective) and, hence, avoid noise; (ii) get a precise classification in terms of data of traces with similar control flow (data-based classification). At runtime, the classification of the historical trace prefixes is used to classify new traces during their execution and predict how they will behave in the future. In particular, the new trace is matched to a cluster, and the corresponding classifier is used to estimate the probability for the trace to achieve a certain outcome. The overall picture of the framework is illustrated in Fig. 1.

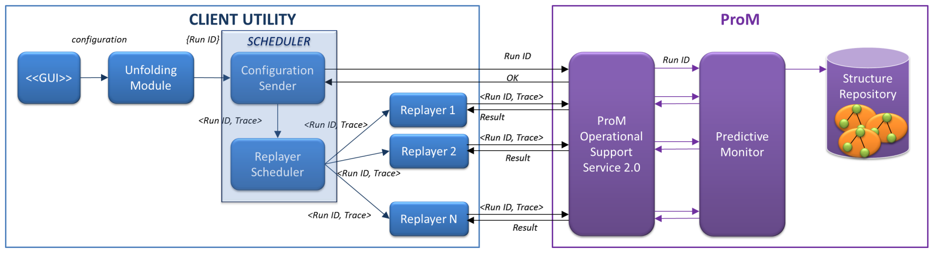


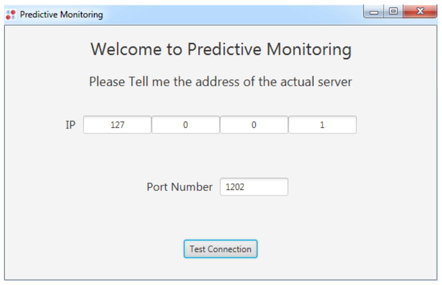
Fig. 2: Logical architecture

The modules of the framework have been implemented by using different techniques for experimentation purposes. The clustering module has been implemented by using two different types of trace encoding and different types of clustering algorithms. In particular, a *frequency based* and a *sequence based* trace encoding approaches have been implemented. The former is realized encoding each execution trace as a vector of event occurrences (on the alphabet of the events), while, in the latter, the trace is encoded as a sequence of events. These encodings can then be passed to the clustering techniques. For instance, the *frequency based* encoding has been used with the *Model-based clustering* [5] and the *sequence based* encoding with the *DBSCAN clustering* [4]. In addition, for model-based clustering, we use the Euclidean distance to identify the clusters while, for DBSCAN, we use the edit distance. Finally, the supervised learning module has been implemented by using decision tree and random forest learning. The possible "instances" of our framework can be obtained through different combinations of these techniques.

The framework has been implemented as an OS provider.[4] Fig. 2 shows the architecture based on the OS. The OS service receives a stream of events (the current execution trace) from a client and forwards it to the OS provider (*Predictive Monitor*) that, based on a repository of historical traces, returns back predictions. The OS service sends these results back to the client. For the implementation of the *Predictive Monitor*, we rely on (i) the Weka implementation of the clustering methods, and on (ii) the WeKa J48 implementation of the C4.5 algorithm and the Weka implementation of random forest for the supervised learning.

As an additional utility, we have implemented a client application providing users with (i) a simple interface for the choice of the (set of) configuration(s), i.e., the selection and the combination of the techniques available in the framework and of the corresponding parameters, to be used for making predictions about the current trace; (ii) a functionality which allows for an extended and fast evaluation of different instances (configurations) of the framework, when a set of testing execution traces (gold standard) is available for evaluation purposes. The client utility presents, indeed, two execution modalities: *prediction*, for the prediction over one or more online execution traces (coming from a workflow engine), and *prediction for evaluation*, returning a set of metrics related to the quality of the results of different configurations (if a testing log is available).

[4] A screencast of this demo can be found at `https://www.dropbox.com/s/yrqszjmvv07okj1/PredictiveMonitoringTool.zip?dl=0`



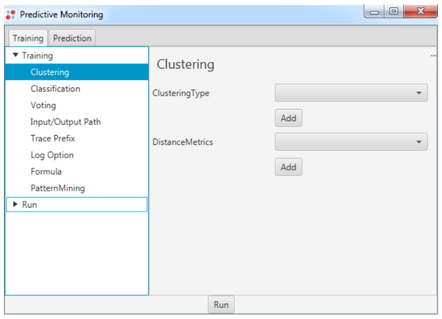
Welcome to Predictive Monitoring

Please Tell me the address of the actual server

IP 127 0 0 1

Port Number 1202

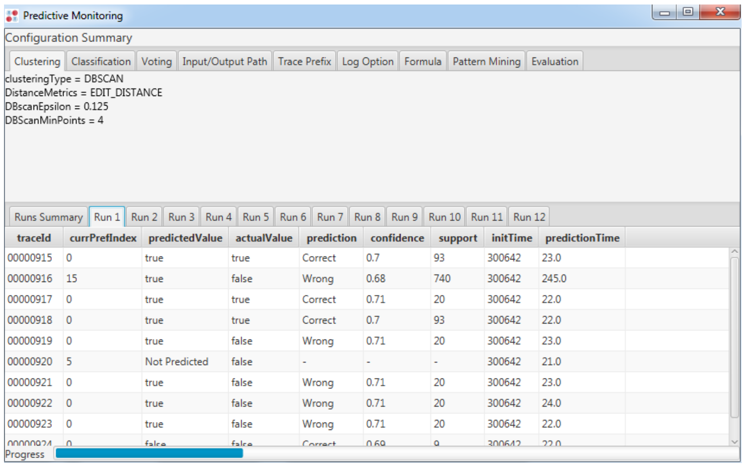
(a) Connection interface



(b) Configuration interface

Fig. 3: Predictive process monitoring client

Fig. 3a shows the starting interface of the client application. Through the GUI the user can select the IP and the port of the OS hosting the Predictive Monitor. Once connected to the server, the user is asked to choose the (set of) configuration(s), i.e., the (combination of) framework clustering and classification techniques and the corresponding parameters, she wants to use (Fig. 3b). By clicking on the button *run*, the configurations are sent to the server.



clusteringType = DBSCAN
DistanceMetrics = EDIT_DISTANCE
DBscanEpsilon = 0.125
DBScanMinPoints = 4

traceId	currPrefIndex	predictedValue	actualValue	prediction	confidence	support	initTime	predictionTime
00000915	0	true	true	Correct	0.7	93	300642	23.0
00000916	15	true	false	Wrong	0.68	740	300642	245.0
00000917	0	true	true	Correct	0.71	20	300642	22.0
00000918	0	true	true	Correct	0.7	93	300642	22.0
00000919	0	true	false	Wrong	0.71	20	300642	23.0
00000920	5	Not Predicted	false	-	-	-	300642	21.0
00000921	0	true	false	Wrong	0.71	20	300642	23.0
00000922	0	true	false	Wrong	0.71	20	300642	24.0
00000923	0	true	false	Wrong	0.71	20	300642	22.0
00000924	0	false	false	Correct	0.69	9	300642	22.0

Fig. 4: Result interface

Fig. 2 shows the logical architecture of the client application and its interactions with the OS Service. As mentioned above, the user can choose whether to use the client just as a "replayer" of a stream of events coming from a workflow engine for prediction purposes or as an evaluation utility for different configurations of the predictive monitoring framework. The *Unfolding Module* combines all the parameters provided by the user into a set of different configuration runs. Here on, each configuration run is associated with an ID (*Run ID*), which will be used to refer such a configuration. The *Configuration Sender* sequentially sends each Run ID to the server that uses it to build the clusters for that specific configuration. As soon as the server has done with the pre-processing, the *Configuration Sender* starts sending the traces to the *Replayer Scheduler* in charge of optimizing the distribution of the traces among different replayers on different threads. Each replayer sends the trace (and the reference to the specific configuration run id) to the server and waits for the results. As soon as the results are provided by the OS Service, they are visualized in the result interface (Fig. 4). Each tab of the result interface refers to a specific configuration run, while the summary tab reports a summary of all the runs.

3 Maturity and Inherence

Predictive Monitoring [6] is an emerging paradigm based on the continuous generation of predictions and recommendations on what activities to perform and what input data values to provide, so that the likelihood to achieve a certain outcome is maximized. Based on an analysis of execution traces, the idea of predictive monitoring is to continuously provide the user with estimations of the likelihood of achieving a certain outcome for a given case. Such predictions generally depend both on: (i) the sequence of activities executed in a given case; and (ii) the values of data attributes after each activity execution in a case.

We have conducted a set of experiments by using the BPI challenge 2011 [1] event log. This log pertains to a healthcare process and, in particular, contains the executions of a process related to the treatment of patients diagnosed with cancer in a large Dutch academic hospital. The performed experiments allowed us to positively answer the following two research questions: (1) "is the framework *effective* in providing *accurate* results as *early* as possible?", and (2) "is the framework *efficient* in providing results?". In addition, we could conclude that the solutions provided by the different instances of the framework offer the possibility to meet different types of needs, by opportunely setting the available configuration parameters. For more information about our experimentation of the tool, the reader is referred to [3].

References

1. 3TU Data Center: BPI Challenge 2011 Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond process mining: From the past to present and future. In: Proc. of CAiSE. pp. 38–52 (2010)
3. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-Based Predictive Process Monitoring. ArXiv e-prints (Jun 2015)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of 2nd International Conference on Knowledge Discovery and. pp. 226–231 (1996)
5. Fraley, C., Raftery, A.E.: Enhanced model-based clustering, density estimation, and discriminant analysis software: MCLUST. Journal of Classification 20, 263–286 (September 2003)
6. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Proceedings of CAiSE 2014 (2014)
7. Westergaard, M., Maggi, F.: Modelling and Verification of a Protocol for Operational Support using Coloured Petri Nets. In: Proc. of ATPN (2011)