Predictive Monitoring of Business Processes

Fabrizio Maria Maggi[1], Chiara Di Francescomarino[2],
Marlon Dumas[1], and Chiara Ghidini[2]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia
{f.m.maggi,marlon.dumas}@ut.ee
[2] FBK-IRST, Via Sommarive 18, 38050 Trento, Italy
{dfmchiara,ghidini}@fbk.eu

Abstract. Modern information systems that support complex business processes generally maintain significant amounts of process execution data, particularly records of events corresponding to the execution of activities (event logs). In this paper, we present an approach to analyze such event logs in order to predictively monitor business constraints during business process execution. At any point during an execution of a process, the user can define business constraints in the form of linear temporal logic rules. When an activity is being executed, the framework identifies input data values that are more (or less) likely to lead to the achievement of each business constraint. Unlike reactive compliance monitoring approaches that detect violations only after they have occurred, our predictive monitoring approach provides early advice so that users can steer ongoing process executions towards the achievement of business constraints. In other words, violations are predicted (and potentially prevented) rather than merely detected. The approach has been implemented in the ProM process mining toolset and validated on a real-life log pertaining to the treatment of cancer patients in a large hospital.

Keywords: Predictive Process Monitoring, Recommendations, Business Constraints, Linear Temporal Logic.

1 Introduction

The execution of business processes is generally subject to internal policies, norms, best practices, regulations, and laws. For example, a doctor may only perform a certain type of surgery if this is preceded by a pre-operational screening, while in a sales process, an order can be archived only after that the customer has confirmed receipt of all ordered items. We use the term *business constraint* to refer a requirement imposed on the execution of a process that separates compliant from non-compliant behavior [20].

Compliance monitoring is an everyday imperative in many organizations. Accordingly, a range of research proposals have addressed the problem of monitoring business processes with respect to business constraints [15,14,16,26,13,19,4,10,5,28]. Given a process model and a set of constraints – expressed, e.g., in temporal logic – these techniques provide a basis to monitor ongoing executions of a process (a.k.a. *cases*) in order to assess whether they comply with the constraints in question. However, these monitoring approaches are *reactive*, in that they allow users to identify a violation only *after*

it has occurred rather than supporting them in *preventing* such violations in the first place.

In this setting, this paper presents a novel monitoring framework, namely *Predictive Business Process Monitoring*, 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 of violation of business constraints is minimized. At any point during the execution of a business process, the user can specify a business constraint using Linear Temporal Logic (LTL). Based on an analysis of execution traces, the framework continuously provides the user with estimations of the likelihood of achieving each business constraint for a given ongoing process execution. The proposed framework takes into account the fact that predictions often 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. For example, for some diseases, doctors may decide whether to perform a surgery or not, based on the age of the patient, while in a sales process, a discount may be applied only for premium customers.

The core of the proposed framework is a method to generate predictions of business constraint fulfillment. Specifically, the technique estimates for each enabled activity in an ongoing case, and for every data input that can be given to this activity, the probability that the execution of the activity with the corresponding data input will lead to the fulfillment of the business constraint. In line with the principle of considering both control-flow and data, the proposed technique proceeds according to a two-phased approach. Given an ongoing case in which certain activities are enabled, we first select from the set of completed execution traces, those that have a prefix "similar" to the (uncompleted) trace of the ongoing case (control-flow matching). Next, for each selected trace, we produce a *data snapshot* consisting of a value assignment for each data attribute up to its matched prefix. Given a business constraint, we classify a data snapshot as a positive or a negative example based on whether the constraint was eventually fulfilled in the completed trace or not. In this way, we map the prediction task to a classification task, wherein the goal is to determine if a given data snapshot leads to a business constraint fulfillment and with what probability. Finally, we solve the resulting classification task using decision tree learning, i.e., we produce a decision tree to discriminate between fulfillments and violations. The decision tree is then used to estimate the probability that the business constraint will be achieved, for each possible combination of input attribute values.

The proposed framework can be applied both for prediction and recommendation. For prediction, the decision tree is used to evaluate the probability for the business constraint to be satisfied for a given combination of attribute values. For recommendation, the decision tree is used to select combinations of attribute values that maximize the probability of the business constraint being satisfied. The predictive monitoring framework has been implemented in the ProM toolset for process mining. The framework has been validated using a real-life log (provided for the 2011 BPI challenge [1]) pertaining to the treatment of cancer patients in a large Dutch academic hospital.

The remainder of the paper is structured as follows. Section 2 introduces a running example. Section 3 introduces concepts pertaining to LTL and decision trees. Section 4 presents the predictive monitoring framework and its implementation. Section 5

discusses the validation on a real-life log. Finally, Section 6 discusses related work and Section 7 draws conclusions and perspectives.

2   Running Example

During the execution of a business process, process participants cooperate to satisfy certain business constraints. At any stage of the process enactment, decisions are taken aimed at achieving the satisfaction of these constraints. Therefore, it becomes crucial for process participants to be provided with predictions on whether the business constraints will be achieved or not and, even more, to receive recommendations about the choices that maximize the probability of satisfying the business constraints.

Fig. 1 shows a BPMN model of a business process we will use as running example. It describes how a patient is nursed according to the instructions of a doctor. During the process execution, the doctor has to make decisions on therapies and on the doses of medicines to be administered to the patient. The process starts when the patient provides the doctor with lab test results. Based on the tests, the doctor formulates a diagnosis. Then, the doctor has to decide the therapy to prescribe. The therapy can be a surgery, a pharmacological therapy or a manipulation. In case of a pharmacological therapy, the doctor has also to prescribe the quantity of medicine the patient has to assume.

In this scenario, historical information about past executions of the process could be used to support the doctor in making decisions by providing him or her with predictions about the (most likely) iter of the disease and recommendations about the best choices to be made in order to guarantee the patient recovery. The approach presented in this paper aims at supporting process participants in their decisions by providing them with predictions about the satisfaction of their constraints and, in case they can influence the process with their decisions, by recommending them the best choices to be made to satisfy their business constraint.

In our example, the constraint the doctor wants to satisfy could be that every diagnosis is eventually followed by the patient recovery. By exploiting data related to the clinical history of other patients with similar characteristics, our technique aims at providing the process participants with predictions about whether the patient will recover or not. In addition, whenever the doctor has to make decisions (e.g., prescribe the type of therapy or choose the dose of a medicine), recommendations are provided about the options for which it is more likely that the patient will recover.
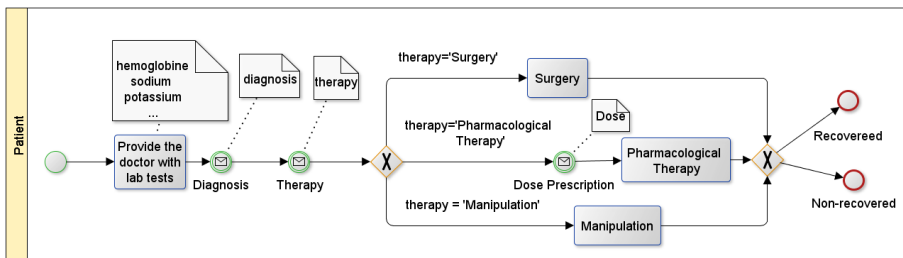

Fig. 1. A simple process describing the medical treatment management

3   Background

In this section, we first introduce the language used for the business constraint definition (LTL), and we then provide an overview on decision tree learning.

3.1   LTL

In our proposed approach, a business constraint can be formulated in terms of LTL rules, as LTL (and its variations) is classically used in the literature for expressing business constraints on procedural knowledge [21]. LTL [23] is a modal logic with modalities devoted to describe time aspects. Classically, LTL is defined for infinite traces. However, when focusing on the compliance of business processes, we use a variant of LTL defined for finite traces (since business process are supposed to complete eventually).

We assume that events occurring during the process execution fall in the set of atomic propositions. LTL rules are constructed from these atoms by applying the temporal operators $\mathbf{X}$ (next), $\mathbf{F}$ (future), $\mathbf{G}$ (globally), and $\mathbf{U}$ (until) in addition to the usual boolean connectives. Given a formula φ, $\mathbf{X}\varphi$ means that the next time instant exists and φ is true in the next time instant (strong next). $\mathbf{F}\varphi$ indicates that φ is true sometimes in the future. $\mathbf{G}\varphi$ means that φ is true always in the future. $\varphi\mathbf{U}\psi$ indicates that φ has to hold at least until ψ holds and ψ must hold in the current or in a future time instant.

In the context of the running example, examples of relevant business constraints formulated in terms of LTL rules include:

- $\varphi_0 = \mathbf{G}(\text{``}diagnosis\text{''} \to \mathbf{F}(\text{``}recovered\text{''}))$,
- $\varphi_1 = \mathbf{F}(\text{``}tumor\ marker\ CA - 19.9\text{''}) \vee \mathbf{F}(\text{``}ca - 125\ using\ meia\text{''})$,
- $\varphi_2 = \mathbf{G}(\text{``}CEA - tumor\ marker\ using\ meia\text{''} \to \mathbf{F}(\text{``}squamous\ cell\ carcinoma\ using\ eia\text{''}))$,
- $\varphi_3 = (\neg\text{``}histological\ examination - biopsies\ nno\text{''})\ \mathbf{U}(\text{``}cytology - ectocervix - \text{''})$,
- $\varphi_4 = \mathbf{F}(\text{``}histological\ examination - big\ resectiep\text{''})$, and
- $\varphi_5 = (\neg\text{``}histological\ examination - biopsies\ nno\text{''})\ \mathbf{U}(\text{``}squamous\ cell\ carcinoma\ using\ eia\text{''})$.

3.2   Decision Tree Learning

Decision tree learning uses a decision tree as a model to predict the value of a target variable based on input variables (features). Decision trees are built from a set of training dataset. Each internal node of the tree is labeled with an input feature. Arcs stemming from a node labeled with a feature are labeled with possible values or value ranges of the feature. Each leaf of the decision tree is labeled with a class, i.e., a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

Each leaf of the decision tree is associated with a class support (*class support*) and a probability distribution (*class probability*). *Class support* represents the number of examples in the training set, that follow the path from the root to the leaf and that are correctly classified; *class probability* (*prob*) is the percentage of examples correctly classified with respect to all the examples following that specific path, as shown in the formula reported in (1).

$$prob = \frac{\#(corr\_class\_leaf\_examples)}{\#(corr\_class\_leaf\_examples + incorr\_class\_leaf\_examples)} \quad (1)$$
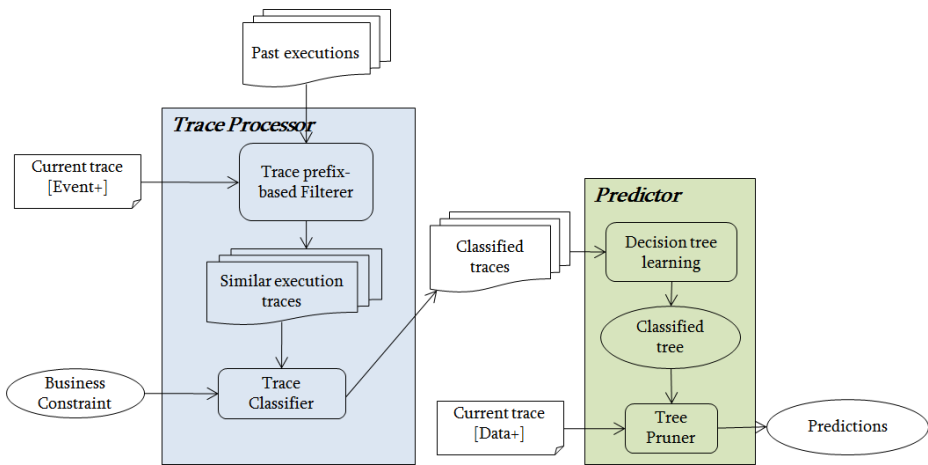
Fig. 2. *Predictive Business Process Monitoring Framework*: architectural overview

One of the most used decision tree learning algorithms is the C4.5 algorithm [24]. C4.5 relies on the normalized information gain to choose, for each node of the tree, the feature to be used for splitting the set of examples. The feature with the highest normalized information gain is chosen to make the decision.

4 Approach

In this section, we present the details of the proposed approach, which combines different existing techniques ranging from clustering approaches to decision tree learning, to provide predictions, at runtime, about the fulfillments of business constraints in an execution trace. In the following sections, we provide an overview of the approach and of the more specific implementation.

4.1 General Approach

Before presenting the approach proposed in this paper, some assumptions should be made. First, we assume that a set of historical execution traces of the process is available from which we can extract information about how the process was executed in the past. Based on the information extracted from the historical traces, we can provide predictions and recommendations for a running execution trace. Second, we assume that the underlying business process should be in some way non-deterministic or, at least, the mechanisms that guide the decisions taken during the process execution should not be known by the user. Any recommendation or prediction would be useless if the process participant already knows how the process develops given the input data values provided (we can think to a doctor who may not know about new therapies, or to a company providing services that does not know about the behaviors of its customers). Third, we assume that data used in the process are globally visible throughout the whole process.

Fig. 2 sketches the proposed *Predictive Business Process Monitoring Framework*. It relies on two main modules: a *Trace Processor* module to filter and classify (past) execution traces and a *Predictor* module, which uses the *Trace Processor* output as training data to provide predictions and recommendations (when an input is requested to the user).

The *Trace prefix-based Filterer* submodule of the *Trace Processor* module extracts from the set of historical traces only those traces having a prefix control flow similar to the one of the current execution trace (up to the current event). The filtering is needed since data values are usually strongly dependent on the control flow path followed by the specific execution. In addition, traces with similar prefixes are more likely to have, eventually in the future, a similar behavior. The similarity between two traces is evaluated based on their edit distance. We use this abstraction (instead of considering traces with a prefix that perfectly matches the current partial trace) to guarantee a sufficient number of examples to be used for the decision tree learning. In particular, a *similarity threshold* can be specified to include more traces in the training set (by considering also the ones that are less similar to the current trace).

Each (historical) trace is identified with a *data snapshot* containing the assignment of values for each attribute in the corresponding selected prefix. The traces (the data snapshots) of the training set are classified by the *Trace Classifier* submodule based on whether, in each of them, the desired business constraint is satisfied or not. The constraint is expressed in terms of a set of LTL formulas. In the case of our running example, the constraint "whenever a diagnosis is performed, then the patient will eventually recover" can be represented in LTL through formula φ_0 reported in Section 3.1.

Formulas have to be satisfied along the whole execution trace. Four possible cases can occur at evaluation time:

- the formula is permanently violated: the prediction is trivial (non-satisfied);
- the formula is permanently satisfied: the prediction is trivial (satisfied);
- the formula is temporary violated/satisfied: the prediction should be able to indicate whether the formula will be satisfied or not in the future.

Once the relevant traces and, therefore, the corresponding data snapshots, are classified, they are passed to the *Decision tree learning* module, in charge to derive the learned decision tree with the associated class support and probability. Fig. 3 shows a decision tree related to our running example: the number of data training examples (with values of the input variables following the path from the root to each leaf) respectively correctly and non-correctly classified is reported on the corresponding leaf of the tree. For example, for values "Joint dislocation" and "Pharmacological therapy", the resulting class is the formula satisfaction ("yes"), with 2 examples of the training set following the same path correctly classified and 1 non-correctly classified, i.e., with a class probability $prob = \frac{2}{2+1} = 0.66$.

All the data values assigned in the past, are supposed to be known by the predictor system at the current execution point of the trace. The tree can hence be pruned by removing all the branches corresponding to known values. The pruning algorithm returns either a unique path (and a unique class) or a subtree of the original tree, according to whether the system is used as predictor (the values of all the tree attributes are known) or as a recommender (there are attributes in the tree that are still unknown), respectively.
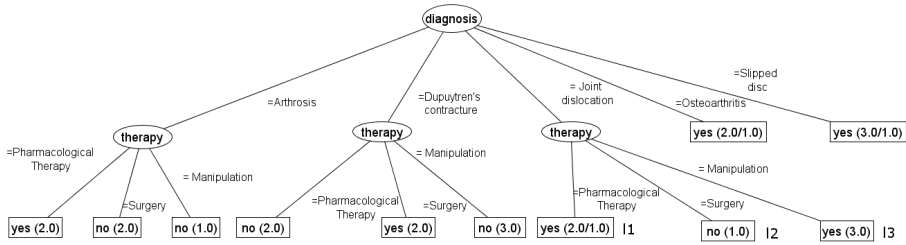
Fig. 3. Example decision tree

In the latter case, leaves are ranked according to the associated class probability. The conditions on the values of the unknown attributes corresponding to the leaves with the highest rankings are returned to the user as recommendations.

For example, consider the case in which a diagnosis ("Joint dislocation") and a therapy ("Pharmacological therapy") have been given by the doctor. The *Predictor* will consider only the path from the root to leaf l_1 (pruning all the other branches) and will predict the satisfaction of the formula with a probability class $prob = 0.66$ (see Fig. 3). We can also consider the case in which a diagnosis has already been made (e.g., "Joint dislocation"), but no therapy has been prescribed yet. Then, all the branches corresponding to other values of the diagnosis attribute (i.e., "Arthrosis", "Dupuytren's contracture", "Osteoarthritis", "Slipped disc") can be pruned. Only the subtree corresponding to the branch "Joint dislocation" is analyzed and, since no other attribute is known, the class probability of each leaf computed. As shown in Fig. 3, the three leaves have the following classes and class probabilities:

- l_1: satisfied with $prob_{l1} = \frac{2}{2+1} = 0.66$
- l_2: non-satisfied with $prob_{l2} = \frac{1}{1} = 1$
- l_3: satisfied with $prob_{l3} = \frac{3}{3+0} = 1$

The system will hence recommend "Manipulation" ($prob_{l3} = 1$).

Note that, if we consider as a feature of the decision tree the next activity to be executed, our framework is also able to recommend which activity should be performed next to maximize the probability of satisfying a business constraint.

4.2 Implementation

The approach has been implemented in the ProM process mining toolset. ProM provides a generic Operational Support (OS) environment [2,29] 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. Our *Predictive Business Process Monitoring Framework* has been implemented as an OS provider.

Fig. 4 shows the entire architecture. The OS service receives a stream of events (including the current execution trace) from a workflow management system and forwards
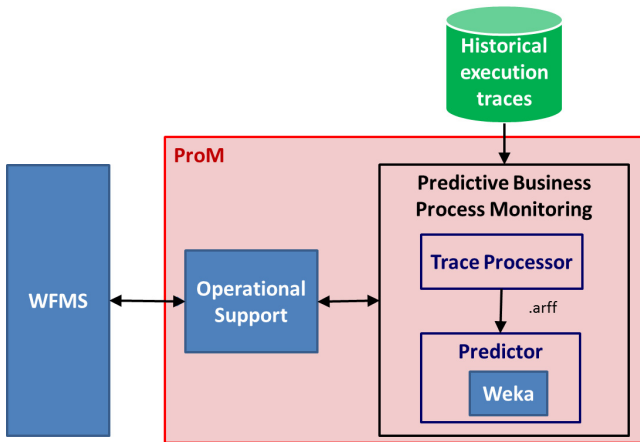
Fig. 4. *Predictive Business Process Monitoring Framework*: implemented architecture

it to the *Predictive Business Process Monitoring Framework* that returns back predictions and recommendations. The OS service sends these results back to the workflow management system.

For the implementation of the *Predictor*, we rely on the WeKa J48 implementation of the C4.5 algorithm, which takes as input a .arff file and builds a decision tree. The .arff file contains a list of typed variables (including the target variable) and, for each trace prefix (i.e., for each data snapshot), the corresponding values. This file is created by the *Trace Processor* and passed to the *Predictor*. The resulting decision tree is then analyzed to generate predictions and recommendations.

5   Experimentation

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 whole event log contains $1,143$ cases and $150,291$ events distributed across 623 event classes (activities). Each case refers to the treatment of a different patient. The event log contains domain specific attributes that are both case attributes and event attributes in addition to the standard XES attributes.[1] For example, *Age*, *Diagnosis*, and *Treatment code* are case attributes and *Activity code*, *Number of executions*, *Specialism code*, and *Group* are event attributes.

In our experimentation, first, we have ordered the traces in the log based on the time at which the first event of each trace has occurred. Then, we have splitted the log in two parts. We have used the first part (80% of the traces) as training set, i.e., we have used these traces as historical data to derive predictions. We have implemented a log replayer to simulate the execution of the remaining traces (remaining 20%) and send them as an event stream to the OS service in ProM (test set).

---

[1] XES (eXtensible Event Stream) is an XML-based standard for event logs proposed by the IEEE Task Force on Process Mining (www.xes-standard.org).
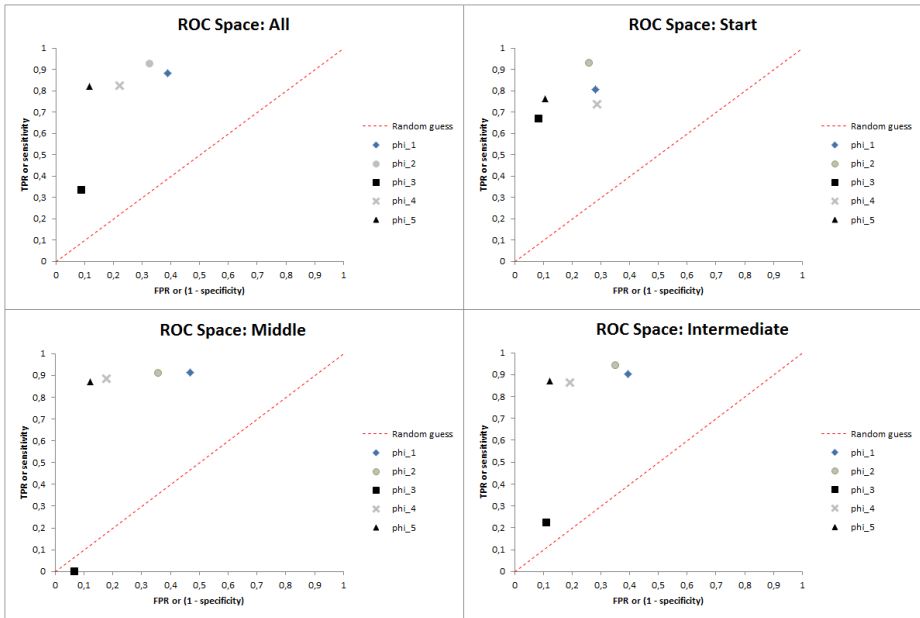
Fig. 5. ROC spaces drawn for different LTL formulas and different evaluation points. Similarity threshold: 0.8; Minimum number of traces: 30.

We defined 5 business constraints corresponding to a subset (from φ_1 to φ_5) of the LTL rules reported in Section 3.1. This set of rules, indeed, allows us to exercise all the LTL constructs while investigating possibly real business constraints. We have asked for a prediction about each of the defined business constraints in different evaluation points during the replay of each trace in our test set. In particular, we have considered as evaluation points the initial event (start event) of each trace, an early event (i.e., an event located at about 1/4 of each trace), and an intermediate event (i.e., an event located in the middle of each trace).

As well as a similarity threshold (see Section 4.1), the implemented OS provider allows the user to specify a minimum number of traces to be used in the training set. In this way, if the threshold does not guarantee a sufficient number of examples, further traces are considered from the set of historical traces with a similarity with the current execution trace lower than the specified threshold. In a first experiment, we have considered a similarity threshold of 0.8 and a minimum number of traces of 30.

For evaluating the effectiveness of our approach, we have used the ROC space analysis. In particular, we have classified predictions in four categories, i.e., *i*) true-positive (T_P: positive outcomes correctly predicted); *ii*) false-positive (F_P: negative outcomes predicted as positive); *iii*) true-negative (T_N: negative outcomes correctly predicted); *iv*) false-negative (F_N: positive outcomes predicted as negative). The *gold standard* used as reference is the set of all true positive instances. In our experiments, we can easily identify the true positive instances. Indeed, if we are asking for a prediction at a certain point in time during the replay of a trace, we can understand if the prediction is correct by replaying the trace until the end.

Table 1. Evaluation of the approach for different LTL formulas, different evaluation points. Similarity threshold: 0.8; Minimum number of traces: 30.

	TP	FP	FN	TN	TPR	FPR	PPV	F1	ACC
φ_1									
Start	46	18	11	46	0.807	0.281	0.718	0.76	**0.76**
Early	73	37	7	42	0.912	0.468	0.663	0.768	**0.723**
Intermediate	75	34	8	52	0.903	0.395	0.688	0.781	**0.751**
All	194	89	26	140	0.881	0.388	0.685	0.771	**0.743**
φ_2									
Start	104	12	8	34	0.928	0.26	0.896	0.912	**0.873**
Early	101	19	10	34	0.909	0.358	0.841	0.874	**0.823**
Intermediate	110	19	7	35	0.94	0.351	0.852	0.894	**0.847**
All	315	50	25	103	0.926	0.326	0.863	0.893	**0.847**
φ_3									
Start	8	13	4	140	0.666	0.084	0.38	0.484	**0.896**
Early	0	11	9	148	0	0.06	0	0	**0.88**
Intermediate	2	18	7	143	0.222	0.111	0.1	0.137	**0.852**
All	10	42	20	431	0.333	0.088	0.192	0.243	**0.876**
φ_4									
Start	53	33	19	82	0.736	0.286	0.616	0.67	**0.721**
Early	54	18	7	83	0.885	0.178	0.75	0.812	**0.845**
Intermediate	57	22	9	92	0.863	0.192	0.721	0.786	**0.827**
All	164	73	35	257	0.824	0.221	0.691	0.752	**0.795**
φ_5									
Start	55	10	17	85	0.763	0.105	0.846	0.802	**0.838**
Early	52	13	11	94	0.825	0.121	0.8	0.812	**0.858**
Intermediate	61	14	9	100	0.871	0.122	0.813	0.841	**0.875**
All	168	37	37	279	0.819	0.117	0.819	0.819	**0.857**

To draw a ROC space, we need two metrics, i.e., the *true positive rate (TPR)*, represented on the y axis, and the *false positive rate (FPR)*, represented on the x axis. The TPR (or recall) defines how many positive outcomes are correctly predicted among all positive examples available:

$$TPR = \frac{T_P}{T_P + F_N}. \tag{2}$$

On the other hand, the FPR defines how many negative outcomes are predicted as positive among all negative examples available:

$$FPR = \frac{F_P}{F_P + T_N}. \tag{3}$$

We have classified predictions for each LTL rule φ_i, and, therefore, each of them is represented as one point in the ROC space. In Fig. 5, we show four spaces drawn by classifying the evaluation points by position (start, early, intermediate). In the figure, we also show the results obtained by considering all the evaluation points together. Note that the best possible prediction method would yield a point in the upper left corner of the ROC space, representing 100% sensitivity (no false negatives) and 100% specificity (no false positives). A completely random guess would give a point along a diagonal line from the left bottom to the top right corners. Points above the diagonal represent good classification results, points below the line poor results.
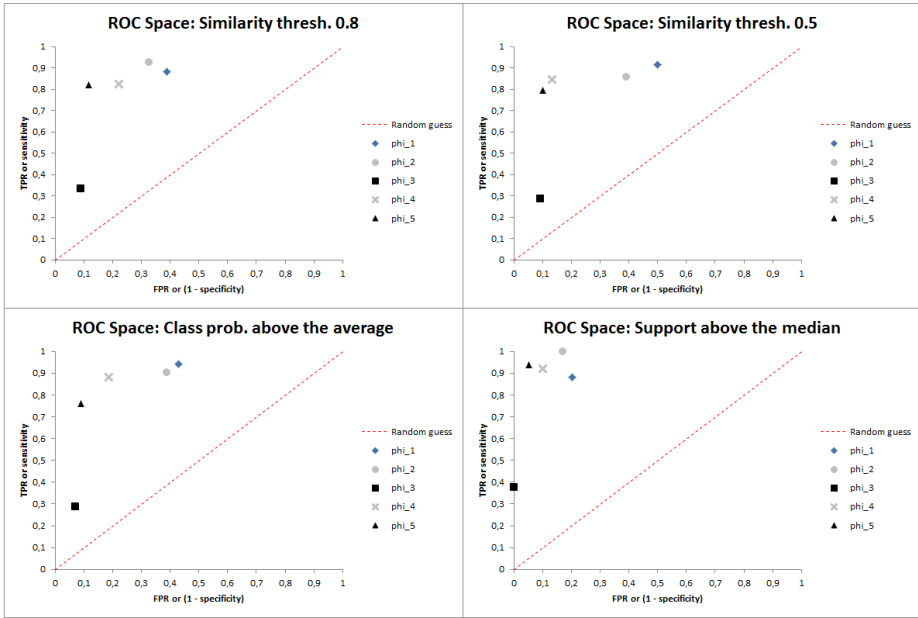
Fig. 6. Comparison of ROC spaces drawn using (1) a similarity threshold of 0.8, (2) a similarity threshold of 0.5, (3) class probability higher than the average, and (4) class support higher than the median of the class supports.

The ROC space analysis highlights that for φ_1, φ_2, φ_4, and φ_5 our OS provider was able to discriminate well between positive and negative outcomes.[2] The results for φ_3 are less good since the number of positive examples for this formula is extremely low and the discovered decision tree overfits.

In general, the position in a trace in which we ask for a prediction does not affect significantly its reliability. In the presented scenario, in which case attributes are available since before the initial event occurs, this is true also for the initial event. Nevertheless, in case of overfitting, there is more variability. Table 1 shows that our results are good also in terms of *positive predictive value (PPV)*, or precision, indicating how many positive outcomes are correctly predicted among all the outcomes predicted as positive:

$$PPV = \frac{T_P}{T_P + F_P}, \tag{4}$$

in terms of *harmonic mean* of precision and recall:

$$F_1 = 2 \cdot \frac{PPV \cdot TPR}{PPV + TPR}, \tag{5}$$

---

[2] Note that, in some cases, the OS provider does not return any prediction. This is due to the fact that, when one of the features reported in the decision tree is an enumeration (and this is the case for several attributes in the considered log), it can happen that not all the possible values of the feature are included in a path from the root to a leaf of the decision tree. Therefore, it is not possible to do any prediction about the behavior of executions in which the feature has one of these values.

Table 2. Evaluation of the approach using (1) a similarity threshold of 0.8, (2) a similarity threshold of 0.5, (3) class probability higher than the average, and (4) class support higher than the median of the class supports

	TPR	FPR	PPV	F1	ACC	LOSS
φ_1						
Similarity thresh. 0.8	0.881	0.388	0.685	0.771	**0.743**	-
Similarity thresh. 0.5	0.915	0.498	0.612	0.734	**0.693**	-
Class prob. above the average	0.94	0.429	0.714	0.812	**0.767**	0.223
Support above the median	0.88	0.201	0.83	0.854	**0.841**	0.508
φ_2						
Similarity thresh. 0.8	0.926	0.326	0.863	0.893	**0.847**	-
Similarity thresh. 0.5	0.858	0.391	0.831	0.844	**0.781**	-
Class prob. above the average	0.903	0.39	0.851	0.876	**0.818**	0.294
Support above the median	1	0.171	0.971	0.985	**0.974**	0.519
φ_3						
Similarity thresh. 0.8	0.333	0.088	0.192	0.243	**0.876**	-
Similarity thresh. 0.5	0.285	0.092	0.188	0.227	**0.864**	-
Class prob. above the average	0.285	0.07	0.176	0.218	**0.897**	0.167
Support above the median	0.375	0	1	0.545	**0.977**	0.559
φ_4						
Similarity thresh. 0.8	0.824	0.221	0.691	0.752	**0.795**	-
Similarity thresh. 0.5	0.846	0.132	0.754	0.797	**0.86**	-
Class prob. above the average	0.881	0.186	0.728	0.797	**0.838**	0.206
Support above the median	0.92	0.1	0.793	0.851	**0.905**	0.518
φ_5						
Similarity thresh. 0.8	0.819	0.117	0.819	0.819	**0.857**	-
Similarity thresh. 0.5	0.794	0.101	0.807	0.801	**0.862**	-
Class prob. above the average	0.761	0.089	0.809	0.784	**0.86**	0.23
Support above the median	0.938	0.053	0.938	0.938	**0.942**	0.53

and in terms of accuracy. Accuracy is particularly important in our context since it indicates how many times a prediction was correct:

$$ACC = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \tag{6}$$

Note that the accuracy value is good also in case of overfitting (formula φ_3).

In a second experiment, we used a lower similarity threshold (0.5) and, again, a minimum number of traces equal to 30. The results for this experiment (for all the evaluation points together) are reported in Table 2 and in Fig. 6. This experiment shows that generating predictions based on a higher number of historical traces not always improves the quality of the results. This is due to the fact that, even if we are considering a larger training set, this set also includes traces that are quite dissimilar from the current trace, thus producing misleading results.

One way of assessing the reliability or "goodness" of a prediction is to use its *class probability*. In Table 2 and in Fig. 6, we show the results obtained by filtering out predictions with a class probability that is lower than the average. Table 2 also reports the prediction loss (LOSS), i.e., the percentage of predictions lost when filtering out predictions with a low class probability. This experiment shows that considering only predictions with a high class probability not always improves the quality of the results, though the percentage of predictions lost is not high (about 20%).

Another way of evaluating the reliability of a prediction is to consider its class sup-port. In Table 2 and in Fig. 6, we show the results obtained by filtering out predictions with support lower than the median of the supports. In this case, although the cut of predictions is high (more than half of the predictions are filtered out), there is a clear improvement in all the considered metrics: in the ROC dimensions, in the F-measure as well as in the average accuracy of the predictions.

In summary, the evaluation shows that the proposed approach is feasible and pro-vides accurate predictions (and hence recommendations). Results seem overall not to be affected by the position of the evaluation point, thus demonstrating that the approach works well even when few variables are known. Support seems to be an important fac-tor influencing the results, i.e., the more evidences we have in the training set, the more accurate are the produced predictions. If on the one hand this highlights the need to have adequate training sets, on the other it also shows that sacrificing outlier predictions, it is possible to obtain very accurate results (accuracy around 0.9).

6   Related Work

In the literature, there are some works that provide approaches for generating predic-tions and recommendations during process execution and are focused on the time per-spective. In [3,2], the authors present a set of approaches based on annotated transition systems containing time information extracted from event logs. The annotated transition systems are used to check time conformance while cases are being executed, predict the remaining processing time of incomplete cases, and recommend appropriate activities to end users working on these cases. In [9], an ad-hoc predictive clustering approach is presented, in which context-related execution scenarios are discovered and modeled through state-aware performance predictors. In [25], the authors introduce a method for predicting the remaining execution time of a process based on stochastic Petri nets.

There are several works focusing on generating predictions and recommendations to reduce risks. For example, in [7], the authors present a technique to support process participants in making risk-informed decisions, with the aim of reducing the process risks. Risks are predicted by traversing decision trees generated from the logs of past process executions. In [22], the authors propose an approach for predicting of time-related process risks by identifying (using statistical principles) indicators observable in event logs that highlight the possibility of transgressing deadlines. In [27], the authors propose an approach for Root Cause Analysis based on classification algorithms. After enriching a log with information like workload, occurrence of delay and involvement of resources, they use decision trees to identify the causes of overtime faults.

An approach for prediction of abnormal termination of business processes has been presented in [11]. Here, a fault detection algorithm (local outlier factor) is used to es-timate the probability of a fault to occur. Alarms are provided to early notify probable abnormal terminations to prevent risks rather than simply react to them. In [6], Castel-lanos et al. present a business operations management platform equipped with time series forecasting functionalities. This platform allows for predictions of metric values on running process instances as well as for predictions of aggregated metric values of future instances (e.g., the number of orders that will be placed next Monday). Predictive

monitoring focusing on failures and quality has also been applied to real case studies (e.g., in transportation contexts [18,8]).

A key difference between these approaches and our technique is that they rely either on the control-flow or on the data perspective for making predictions at runtime, whereas we take both perspectives into consideration. In addition, the purpose of our recommendations is different. We provide recommendations neither to reduce risks nor to satisfy/discover timing constraints. We aim instead at maximizing the likelihood of satisfying business constraints expressed in the form of LTL rules.

7   Conclusion

This paper presented a framework for predictive business process monitoring based on the estimation of probabilities of fulfillment of LTL rules at different points during the execution of a case. The framework takes into account both the sequencing of activities as well as data associated to the execution of each activity. A validation of the framework using a real-life log demonstrates that recommendations generated based on the framework have a promising level of accuracy when sufficient support is available.

Increased accuracy could be achieved by extending the technique along two directions. First, the proposed technique matches the trace of an ongoing case against prefixes of completed traces based on edit distance. While this is a well-known measure of similarity and suitable as a first step in this study, other approaches could be considered, including trace similarity measures based on occurrences of n-grams, counts of activities and activity pairs, and other relevant features that have been studied in the context of trace clustering [17]. In a similar vein, discriminative sequence mining techniques [12] could be applied in order to extract prefix patterns that are associated with fulfillment of a given business constraint. These patterns can also be taken as input in the prediction. Secondly, we have considered the use of decision trees to build the classifier. With larger number of attributes, which might be encountered in richer logs, decision trees are likely to exhibit lower accuracy due to their inherent weaknesses when dealing with large feature sets. In this context, other classification techniques, such as random forests or sparse logistic regression are possible alternatives.

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: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 38–52. Springer, Heidelberg (2010)

3. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. 36(2), 450–475 (2011)
4. Beheshti, S.-M.-R., Benatallah, B., Motahari-Nezhad, H.R., Sakr, S.: A query language for analyzing business processes execution. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 281–297. Springer, Heidelberg (2011)
5. Birukou, A., D'Andrea, V., Leymann, F., Serafinski, J., Silveira, P., Strauch, S., Tluczek, M.: An integrated solution for runtime compliance governance in SOA. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 122–136. Springer, Heidelberg (2010)
6. Castellanos, M., Salazar, N., Casati, F., Dayal, U., Shan, M.-C.: Predictive business operations management. In: Bhalla, S. (ed.) DNIS 2005. LNCS, vol. 3433, pp. 1–14. Springer, Heidelberg (2005)
7. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013)
8. Feldman, Z., Fournier, F., Franklin, R., Metzger, A.: Proactive event processing in action: a case study on the proactive management of transport processes. In: DEBS (2013)
9. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
10. Holmes, T., Mulo, E., Zdun, U., Dustdar, S.: Model-aware monitoring of SOAs for compliance. In: Service Engineering, pp. 117–136. Springer (2011)
11. Kang, B., Kim, D., Kang, S.H.: Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction. Expert Syst. Appl. (2012)
12. Lo, D., Cheng, H.: Lucia: Mining closed discriminative dyadic sequential patterns. In: Proc. of EDBT, pp. 21–32. Springer (2011)
13. Ly, L.T., Rinderle-Ma, S., Knuplesch, D., Dadam, P.: Monitoring business process compliance using compliance rule graphs. In: Meersman, R., Dillon, T., Herrero, P., Kumar, A., Reichert, M., Qing, L., Ooi, B.-C., Damiani, E., Schmidt, D.C., White, J., Hauswirth, M., Hitzler, P., Mohania, M. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 82–99. Springer, Heidelberg (2011)
14. Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P.: Runtime verification of LTL-based declarative process models. In: Khurshid, S., Sen, K. (eds.) RV 2011. LNCS, vol. 7186, pp. 131–146. Springer, Heidelberg (2012)
15. Maggi, F.M., Montali, M., Westergaard, M., van der Aalst, W.M.P.: Monitoring business constraints with linear temporal logic: An approach based on colored automata. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 132–147. Springer, Heidelberg (2011)
16. Maggi, F.M., Montali, M., van der Aalst, W.M.P.: An operational decision support framework for monitoring business constraints. In: de Lara, J., Zisman, A. (eds.) Fundamental Approaches to Software Engineering. LNCS, vol. 7212, pp. 146–162. Springer, Heidelberg (2012)
17. de Medeiros, A.K.A., Guzzo, A., Greco, G., van der Aalst, W.M.P., Weijters, A.J.M.M.T., van Dongen, B.F., Saccà, D.: Process mining based on clustering: A quest for precision. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 17–29. Springer, Heidelberg (2008)
18. Metzger, A., Franklin, R., Engel, Y.: Predictive monitoring of heterogeneous service-oriented business networks: The transport and logistics case. In: SRII (2012)
19. Montali, M., Pesic, M., van der Aalst, W.M.P., Chesani, F., Mello, P., Storari, S.: Declarative specification and verification of service choreographiess. TWEB 4(1) (2010)

20. Pesic, M., van der Aalst, W.M.P.: A Declarative Approach for Flexible Business Processes Management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)

21. Pesic, M., Schonenberg, H., van der Aalst, W.M.P.: Declare: Full support for loosely-structured processes. In: Proc. of EDOC, pp. 287–300 (2007)

22. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting deadline transgressions using event logs. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 211–216. Springer, Heidelberg (2013)

23. Pnueli, A.: The temporal logic of programs. In: SFCS, pp. 46–57 (1977)

24. Quinlan, J.R.: C4.5: Programs for Machine Learning. M. Kaufmann Publishers Inc (1993)

25. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013)

26. Santos, E.A.P., Francisco, R., Vieira, A.D.: F.R. Loures, E., Busetti, M.A.: Modeling business rules for supervisory control of process-aware information systems (2012)

27. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. Lecture Notes in Business Information Processing, vol. 132, pp. 174–186. Springer, Heidelberg (2013)

28. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-based monitoring of process execution violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 182–198. Springer, Heidelberg (2011)

29. Westergaard, M., Maggi, F.M.: Modeling and verification of a protocol for operational support using coloured petri nets. In: Kristensen, L.M., Petrucci, L. (eds.) PETRI NETS 2011. LNCS, vol. 6709, pp. 169–188. Springer, Heidelberg (2011)