Classification Techniques for Conformance and Performance Checking in Process Analysis

Hind Chfouka[1], Andrea Corradini[1] and Roberto Guanciale[2]

[1] Department of Computer Science, University of Pisa, Italy
[2] School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden

Abstract. Standard process analysis techniques, like conformance checking or performance evaluation, are enabled by the existence of event logs that trace the process executions and by the presence of a model that formally represents the process. Such analysis techniques use only part of the huge amount of data recorded in event logs. In this paper the goal is to exploit this data to extract useful information for conformance checking and performance analysis. We present an approach that using standard classification technique, explores how data influence process behaviors by affecting its conformance or performance.

1 Introduction

Today, many Information Systems that support the concept of business process record all events occurring during the process execution in event logs. An event log represents a trace of the process behavior that can be observed and analyzed in order to tune it with the business objectives pursued. Process Mining provides a set of techniques for process discovery and analysis. Through the knowledge extracted from event logs, process discovery allows the construction of a process model. Process analysis instead assumes the existence of a model and it consists of checking the conformance and performance of the process executions with respect to it.

The motivation of this paper is based on the observation that existing event logs are rich of data, and this data is used only in part by the existing process analysis techniques. Since data is considered a valid source of knowledge, developing a technique based on data analysis can provide advantages for process analysis. Therefore, our goal consists of finding a way for transforming the data recorded during the process activities in knowledge useful for the analysis. To this aim, one possible approach is to try to understand how data may influence the process behavior. In particular, it is interesting to understand how may the data influence the conformance and performance of the process executions. Exploring this influence can provide qualitative information about the causes of possible anomalies in the process behavior, facilitating the task of taking corrective measures.

In order to analyze data arising from event logs, in this paper we present an approach based on *data mining* [**?**]. In particular, we exploit *classification* techniques to find patterns on data in presence of which conformance errors occur. The same approach has been extended to evaluate how data influences process efficiency, by combining classification problems with performance evaluation: this is just summarized briefly in the paper due to size constraints. Classification is a well-known *machine learning* technique, and it consists of identifying to which category, among a given set, a new observation belongs. This is done on the basis of a training set of data containing observations whose category membership is known.

After the presentation of a case study in Section 2, Section 3 introduces some background concepts about process analysis. First the algorithm log replay is briefly described, then conformance analysis is presented. This analysis is illustrated through the business process of Section 2. In Section 4 our approach to process analysis based on classification is presented. First a few basic concepts about the classification technique are described, next the conformance analysis based on classification is presented in detail through an illustration with our case study, moreover a possible extension of the approach for performance analysis is presented. Finally, Section 5 describes the implementation of our approach provided as *ProM6*[**?**] plug-ins, as well the methodology to follow in the process analysis based on classification.

2 The case study: a sale business process

A business process can be seen as a collection of activities occurring within an organization that lead to a specific goal. There are several formalisms for representing business processes. In this paper we represent them using Petri nets [**?**][**?**] since this make possible exploiting some existing process analysis techniques that are based on Petri nets. However, in the business management context, Petri nets can be not expressive enough and usually process models are presented as flowcharts, in particular using the standard for the business process modeling: BPMN (Business Process Model and Notation) [**?**]. Transforming a BPMN model into a Petri net is possible thanks to a mapping technique presented in [**?**].

Figure 1 presents the Petri net model for a sale process [3]. This business process abstractly represents a fragment of the procedure followed in a commercial organization for managing orders received from clients. In general this procedure includes several activities and each of them involves a specific department of the organization. We assume that the sale process begins with the notification of an order from a client, so the first activity is simply called *Order*. After the order is

[3] The Petri net of Figure 1 is obtained by transforming a BPMN model using the algorithm of [**?**], and mapping each activity to a pair of start/end transitions. The resulting net has been simplified by removing the unnecessary *invisible transitions* used to encode the *Join* or *Fork* gateway [**?**].

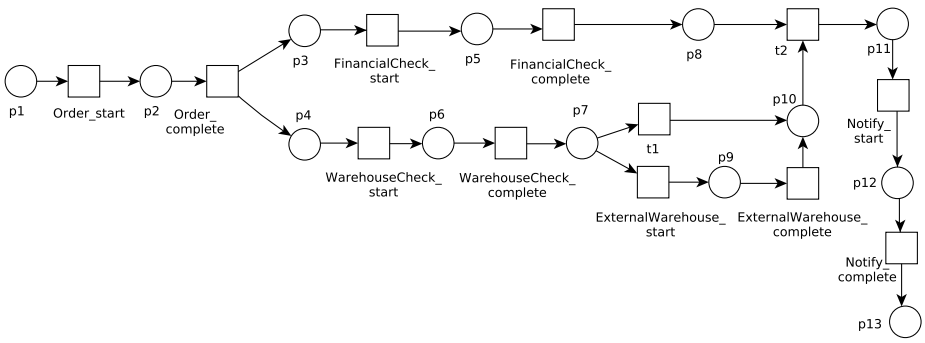

Fig. 1. Petri net for a a sale process

received, the sale process continues with some activities that can be done in parallel, since they do not present any dependencies. The *FinancialCheck* activity represents the financial analysis of the order (i.e verifying the financial situation of the client). Concurrently, the *WarehouseCheck* activity checks the availability of merchandise requested by the order and, in case of a shortage, starts a supplying procedure. This is done through the activity *ExternalWarehouse*. After the completion of the parallel activities, a synchronization is needed before continuing the sale procedure with the final activity called *Notify*, by which a result regarding the acceptance of the order is communicated to the client.

3 Process Analysis

Process analysis is performed using the Petri net representing the process and an event log recording the related executions, also called *process instances*. The basic building blocks of event logs are events. An event e can be seen as a pair $e = (a, t)$ representing an action a recorded and the corresponding timestamp t. Events that belong to the same process are grouped into *traces*. A trace T is a finite sequence of events ordered based on timestamp. A log L is a set of traces, recording the activities performed by a system during a finite number of process executions. In this paper the following assumptions are made:

- All traces are instances of the same process.
- For each action there exist two corresponding transitions in the net that will be denoted, for simplicity, by the same name of the action.

3.1 Log replay algorithm

The key algorithm exploited to analyze a Petri net model with respect to the log is the *log replay* algorithm [**?**][**?**][**?**]. Given a Petri net model and an event log as input to the algorithm, the output results can be used to check the conformance

of traces and to evaluate some performance metrics. For each trace in the log, the algorithm starts by placing one token in the start place of the net. For each event in the trace the corresponding transition is fired assuming a *non-blocking* behavior of the algorithm, then the marking of the net is updated. Non-blocking replay means that whenever a firing of a disabled transition is needed, the algorithm enables the transition either creating artificial tokens in the pre-set or, if possible, firing some invisible transitions; the non-determinism in this procedure is resolved with a suitable cost function.

3.2 Conformance Analysis

The goal of the conformance analysis is to check if a trace complies with the Petri net modeling the business process. Conformance problems can be discovered by analyzing tokens artificially created during the replay *(the missing tokens)* and tokens not consumed *(the remaining tokens)*. Figure 2 presents two different traces, T and T', of the business process presented in Section 2. The log replay of the trace T, which is compliant with the Petri net, terminates with a marking containing one token in the end place $\{p13 \rightarrow 1\}$ and without reporting any missing tokens.

T	Order #1 start 9:30:50	Order #1 complete 10:15:00	WarhouseCheck #1 start 10:35:25	FinancialCheck #1 start 10:40:20	FinancialCheck #1 complete 12:00:20	WarhouseCheck #1 complete 12:40:20	Notify #1 start 12:55:20	Notify #1 complete 13:00:10

T'	Order #1 start 9:30:50	Order #1 complete 10:15:00	WarhouseCheck #1 start 10:35:25	FinancialCheck #1 start 10:40:20	WarhouseCheck #1 complete 12:40:20	Notify #1 start 12:55:20	Notify #1 complete 13:00:10

Fig. 2. Conform (T) and non-conform (T') traces

The log replay of the trace T' terminates with remaining tokens $\{p5 \rightarrow 1$, $p13 \rightarrow 1\}$ and one missing token $\{p8 \rightarrow 1\}$. The missing token is created artificially and this fact witness a wrong execution of the event $Notify\_start$. In fact, in T' this event is executed before the termination of the activity *FinancialCheck*: this is interpreted as a non-conformance to the process model.

4 An approach based on classification for Process Analysis

Employing data mining techniques for process analysis is encouraged by the presence of a huge amount of data recorded in event logs under *attribute* format. The implicit information contained in those data could be significant for the process behavior analysis. In fact, this potential information could contain an explanation for the deviations discovered during conformance analysis, and for the performance level provided by the process. For this reason, we are interested in discovering how the process data may influence the conformance and the performance of the process executions. This is done with an approach based on

classification, a classical data mining technique able to detect patterns on data in correspondence to which the process assumes a specific behavior.

4.1 Classification: basic concepts

In a classification problem [**?**], data is represented by a collection of records (called instances), and each of them is characterized by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is a set of attributes and y is a special attribute called *target attribute*. The target denotes the class to which a record belongs. The goal is to learn a classification model that maps each attribute set $\mathbf{x}$ to one of the predefined class labels y.

A classification model can be useful for a descriptive purpose: it provides an explanatory tool to distinguish between objects of different classes, and for a predictive purpose: the model can be used to predict the class label of new records. In this way the model acts as a black box that automatically assigns a class label to an attribute set of a new record.

There is a general approach to solve a classification problem from a dataset, that is independent from the specific classification model. Each classification technique employs a learning algorithm to identify a model that best fit the relationship between the attribute set and class label of the input data. The model generated should both fit the input data well, and correctly predict the class labels of new records. Finding the right trade-off between this two purposes is the most delicate part of a classification technique, and in general of any machine learning task.

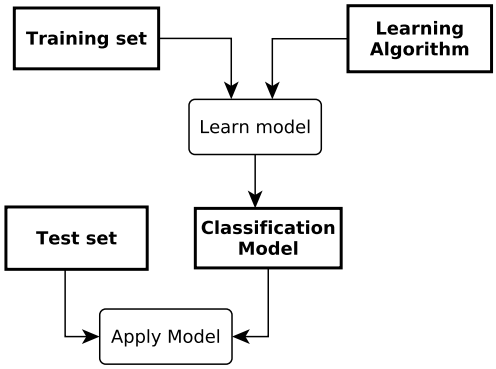

Fig. 3. General approach for solving a classification problem.

Figure 3 summarizes the general approach for solving a classification problem [**?**]. A *Training set* consisting of records whose class labels are known must be provided to a learning algorithm, and it is used to build a classification model. This model is then applied to a *test set*, which consists of records with unknown

class labels. This step allows to estimate the *model accuracy* using one of the standard approaches: e.g. the accuracy can be represented by the fraction of number the records correctly predicted and the total number of records in the test set.

Many classification models can be applied to a classification problem. For our purpose we choose the *decision tree* model since the rules detected by the classification are shown explicitly, providing an overview of how process data affects the process behavior. Furthermore, this choice is encouraged by the simplicity of decision trees. In a decision tree, a class label (possible value of the attribute target) is assigned to each leaf node. The non-terminal nodes, which include the root and the internal nodes, contain attribute test conditions to filter records that have different characteristics.

4.2 Classification for Conformance Checking

In order to identify the possible causes of non-conformance, a classification problem can be formulated. The classification dataset is extracted from the process data: for each process instance, relevant information is extract and included as a record in the dataset. For the process presented in Section 2, a record is characterized by the following attributes: an instance identifier, a client identifier, the client typology (new or consolidated client), the sales manager responsible for the order, the financial officer who conducts the financial evaluation activities, the warehouseman responsible for the warehouse checking, the supplying responsible name in case of a provision, and finally the result communicated to the client for the order issued. The conformance result for the process instance provides an additional and important data, which takes the role of the target attribute. All these attributes are discrete and contribute to build the dataset presented in table 1 of the classification problem. Notice that, since our goal is to illustrate the application of existing data mining techniques to process analysis rather then the evaluation of a new data mining algorithm, we use synthetic data that are generated to emulate a realistic process executions.

In order to build the classification problem, the attributes characterizing the process are extracted from the log, while the conformance attribute is computed using the conformance checking algorithm. The Petri net in Figure 4 summarizes the conformance analysis executed on the event log: edges are labeled with the number of activations done, places are labeled with the number of remaining and missing tokens.

The resulting data highlight that there are 37 out of 250 process executions with conformance errors. The presence of 37 missing tokens in $p11$ represents a forced execution of transition *Notify_start*. In order to provide an interpretation for this results, we have to take into account the non-blocking behavior of the log replay algorithm. In this case, the 37 missing tokens in $p11$ represents the number of times that that the replay the algorithm failed to mimic the event *Notify_start* since the transition associated to the event was not enabled. Moreover, 37 remaining tokens in place $p3$ indicates the number of times that the instances

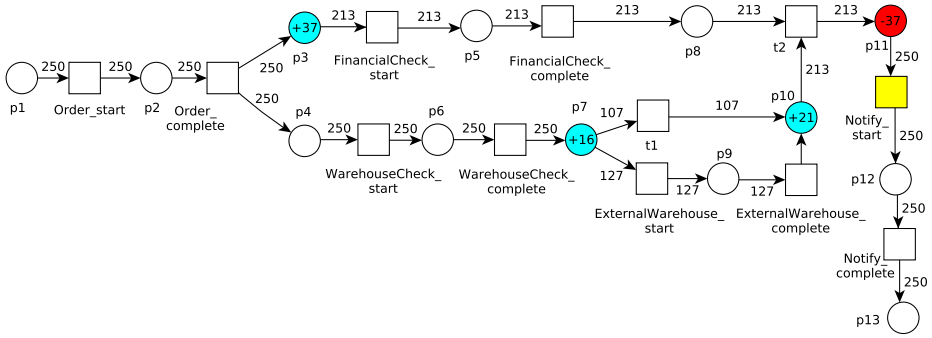

Fig. 4. Petri net: conformance results.

financial activities were not performed. Taking into account the application scenario, we can conclude that 37 instances of the process are not compliant with sale policy since they did not executed the financial activity.

OrdIde	CltIde	CltType	SalMan	FinOff	WrhsMan	SupplyResp	OrdResut	Conf
1	20	consolidate	Marco	Mary	Alex	Gianni	positive	no
2	15	new	Anna	Johann	Roberto	Mario	positive	yes
3	10	consolidate	Maria	Mary	Alessio	Gianni	negative	no
10	18	consolidate	Johann	Mary	Roberto	Gianni	positive	yes
...	...	...	...	...	...	...	....	...

Table 1. Dataset for the conformance analysis.

The results of the conformance checking enable the usage of the existing data mining tools, allowing us to mine the decision tree in Figure 5 as classification model for the sale process. The resulting decision tree describes a data pattern in correspondence to which a process instance could present conformance errors: order managed by the sales manager *Mary* and received from consolidate clients of the organization may not respect the standard sales procedure. To get more significant information for our analysis, From the analysis done for the sales process based on both classification and log replay results, a new scenario of the sale procedure has emerged: orders done by the consolidate clients of the organization are not checked from a financial point of view. This fact can be accepted as a valid behavior in a sale procedure, consequently an extension of the business process model is needed to include the new scenario (*model extension activity*). Alternatively, the misbehaving scenario can be considered as an anomaly in the sales procedure. In this case, since the analysis provided accurate information about the conformance error, corrective measures can be taken in order to avoid errors in the subsequent orders.

The classifier constructed with this approach can be used also in a predictive way. Given a trace recording a process execution the decision tree can be used

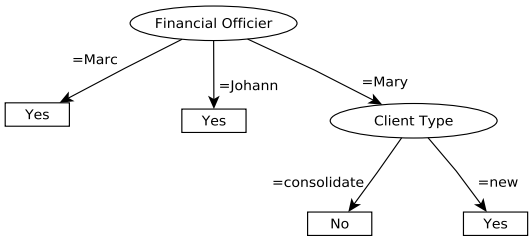

Fig. 5. Decision tree for the conformance analysis.

to predict the conformance result. This bring an advantage in terms of the time required by the analysis since the log replay algorithm takes more time than execution of a decision tree. Moreover, it is worth noting that whenever the set of attributes needed for the classification is known before the completion of a process execution, the non-conformances predicted by the decision tree can be used to promptly alert the stakeholders and activate proper countermeasures.

4.3 Extension of the approach for Performance Analysis

Performance analysis of a process can be carried exploiting log replay. Since a log contains timestamps, during the replay of a trace it is possible to compute (for each place of the Petri net model) some performance measures such as: *synchronization time* (i.e. time interval between arrival of a token in the place and enabling of a transition in the post-set of the place), *sejour time* (i.e. the time interval between arrival and departure of tokens) and *waiting time* (i.e. the time interval between enabling of a transition in the post-set of the place and token departure).

Performing an analysis based on the measures computed by log replay gives just a *quantitative* information about the process performance. In order to understand the *causes* of performance anomalies that can affect the process behavior, one can explore the process data. Discovering how data attribute can influence process performance provides useful information in analyzing and optimizing the process services. For example, discovering data patterns in correspondence to which some activities need more time for completion than others, helps in making decisions about resources distribution to the process activities or in scheduling activities. In addition to the completion time, one could analyze a process under more complex performance metrics such as the synchronization time. A process with parallel branches and synchronizations can present bottleneck activities that lead to increase the execution time of other activities and consequently the completion time of the entire process. To find out the possible data influences on synchronization time, the approach presented in Section 4.2 for conformance analysis through classification can be easily extended to performance analysis. For example, a classification problem can be formulated for each

synchronization point of the process. In this way, the classifier obtained in correspondence to a synchronization point classifies the process instances based on their attribute value and regarding the synchronization time of the point taken into exam.

5 Implementation with ProM

In order to experiment with some business process prototypes, the approach presented in this paper has been implemented as a set of plug-ins that integrates *ProM6* [**?**], an open-source framework implementing Process Mining tools, and *Weka* [**?**], a data mining framework providing machine learning tools. We can classify the plug-ins developed in three categories. The first one includes plug-ins for building the dataset needed for the classification:

- `Generate Instances With Conformance`: this plug-in takes as input an event log and its conformance results computed by log replay. It returns as output a training set as shown in Table 1 in the format needed by Weka classification tool. Note that the plug-in extracts *all* the attributes data recorded in the event log. In case, the dataset resulting can be subject to some *feature selection* techniques before using it as a training set for the classification task.
- `Generate Instances to Classify`: given an event log, the plug-in generates a dataset of instances that can be classified using a classification model.

The second category includes plug-ins for the classifier generation and utilization as a predictive model:

- `Generate Classifier`: given a training set, the plug-in generates a decision tree model using $J48$ algorithm, an implementation of the algorithm $C4.5$ developed by Weka.
- `Classify Instances`: given a set of non-classified instances and a classifier model, this plug-in simply classify the instances according to the classifier.

Finally, the last category includes a set of plug-ins useful for the serialization, deserialization and visualization of the resources used by the previous plug-ins, such as dataset and classifiers.

5.1 Methodology

This last paragraph is dedicated to a description of the methodology to be followed to analyze a business process under the approach presented in this paper.

It should be clear that the analysis presented on the previous Section assumes the presence of a Petri net model of the business process to be analyzed. The model can be expressed in BPMN language as well, but in this case a transformation into a Petri net can be done through the techniques described in [**?**]. The classification analysis needs also an event log tracing the process behavior over

multiple instances. Given a process model M and an event log L, the methodology to be followed for the conformance analysis with the approach presented in Section 4.2 can be summarized by these steps:

1. **Log replay analysis**: this phase consists in an execution of the log replay with the model M and the log L as inputs. The results consists in two structures characterizing both the conformance and the performance of the traces in L. Let us call them $ConfResult$, and $PerfResult$.
2. **Training set construction**: this phase aims to build the training set for the classification technique. This is done through an execution of the plug-in `Generate Instances With Conformance` with the event log L and the conformance result $ConfResult$ as inputs. Let us call the training set obtaind TS.
3. **Classifier building**: in this step the classification model (decision tree) for the conformance analysis is built using the plug-in `Generate Classifier` with the training set TS as input. Let us call the decision tree obtained DT.
4. **Using the classifier**: in this step the goal is to exploit DT and the log replay result $ConfResult$ to establish the corrective measures in case of anomalies. Note that both DT and $ConfResult$ can be visualized as in Figures 5 and 4 into ProM6 thanks to the plug-ins for resources managing that integrate the framework. DT can also be used in a predictive way to establish the conformance of new process instances without needing a log replay execution. Given an event log containing new process instances, a dataset can be built through the plug-in `Generate Instances to Classify` and then classified by the plug-in `Classify Instances` according to the model DT.

To perform a performance analysis based on the extension presented in Section 4.3, the same methodology can be followed modifying the step 2. The construction of the training set is done through a plug-in (analogous to `Generate Instances With Conformance`) that takes as input $PerfResult$ obtained by the step 1 and an indication about the synchronization point chosen for the analysis.

6 Conclusions and Future Work

We presented some preliminary results of a research activity aimed at applying machine learning techniques, i.e. classification algorithms, in the realm of process analysis. The idea is to try to identify some correlation between the data value of certain variables in the individual cases of event logs, and the results of conformance checking. A successful identification of such values could allow to predict the conformance result of a case during execution. The same technique was applied to performance checking as well, but only briefly summarized here because of space constraints. The approach is implemented with a combination of plug-ins of the Process Mining framework ProM and of the classification engine WEKA. Some guidelines about their use are presented in a concluding section. The approach has been successfully tested on synthetic event logs only. An obvious continuation of this work includes experimentations with real event logs:

this will allow us to evaluate the scalability of the approach and its robustness in presence of noise.

Classification techniques in the Process Mining field have already been explored in [**?**] and more recently in [**?**]. In those papers the idea is to explore how data influences the case routing of process flow execution, by assigning a classification problem with each decision point in the model. In particular, [**?**] improves significantly the results of [**?**] by exploiting some recently developed *alignment techniques* between event log cases and the process model. We intend to explore how far these techniques can be applied fruitfully in our framework as well.

References

1. R. A, V. D. Aalst, and W. M. P. Decision mining in prom. *Business Process Management*, 2006.
2. W. V. D. Aalst. The application of petri nets to workflow management, 1998.
3. M. L. G. at University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/, 2011.
4. R. Bruni, A. Corradini, G. Ferrari, T. Flagella, R. Guanciale, and G. Spagnolo. Applying process analysis to the italian egovernment enterprise architecture. In M. Carbone and J.-M. Petit, editors, *Web Services and Formal Methods*, volume 7176 of *Lecture Notes in Computer Science*, pages 111–127. Springer Berlin Heidelberg, 2012.
5. R. Bruni, A. Corradini, G. Ferrari, T. Flagella, R. Guanciale, and G. O. Spagnolo. Misurare processi di business. *Congresso AICA*, 2011.
6. J. Muñoz-Gama and J. Carmona. A fresh look at precision in process conformance. In R. Hull, J. Mendling, and S. Tai, editors, *Business Process Management*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer Berlin Heidelberg, 2010.
7. OMG. http://www.bpmn.org/, 2011.
8. C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *IFIP Congress*, pages 386–390, 1962.
9. E. T. U. Process Mining Group. http://www.processmining.org/prom/start, 2010.
10. A. Rozinat and W. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64 – 95, 2008.
11. P. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining.* Pearson Addison Wesley, 2006.
12. W. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 2(2):182–192, Mar. 2012.