Classification Techniques for Conformance and Performance checking in Process Analysis

Hind Chfouka, Andrea Corradini, Roberto Guanciale

Department of Computer Science, Università di Pisa

Abstract. Standard process analysis techniques like conformance checking or performance evaluation are enabled by the 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 the event logs. In this paper the goal is to transform this data into useful information for the conformance checking and the performance analysis. Through classification technique, we present an approach to investigate how the process data influences its behavior in term of conformance and performance.

1 Introduction

Today, many Information Systems that integrate the concept of business process record all events occurring during the process execution in an event log. 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. Therefor, 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, and this will facilitate 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* [11]. In particular, we exploit *classification* techniques to find patterns on data in presence of which conformance errors occur. Next, the same approach is extended to consider the process behavior in terms of performance to discover how data influences the process efficiency. 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.

Classification techniques in the Process Mining field have already been explored. In [1] the idea is exploring how data influences the case routing of process flow execution. In this contribution the goal is to strengthen the use of data analysis techniques for the benefit of process analysis. This is done by studying how data may influence, in general, the overall process behavior.

After the presentation of a simple business process in Section 2, Section 3 introduces some background concepts about process analysis. First the algorithm log replay is briefly described, then conformance and performance analysis are presented. Both kind of analysis are 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 and performance analysis based on classification are presented illustrating them with the process of Section 2. Finally, Section 5 describes the extension of *ProM6* [9] framework by some plug-ins realizing the approach presented in Section 4.

2 An example of business process model for a sale 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 [8][2] 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) [7]. Transforming a BPMN model into a Petri net is possible thanks to a mapping technique presented in [4][5].

Figure 1 presents the Petri net model for a sale process. This is the business process representing in an abstract way a fragment of the procedure followed in a commercial organization for managing orders received from clients. In general this procedure includes different activities and each of them involves a specific department of the organization. We assume that the sale process begins with receiving an order notification from a client, so the first activity is simply called *Order*. After the order is received, the sale process continues with some activities that can be done in parallel, since they do not present any dependencies. One of these is the *FinancialCheck* activity during which the order is analyzed from a financial point of view (i.e verifying the financial situation of the client). In parallel to this, an activity called *WarehouseCheck* is performed. In this phase

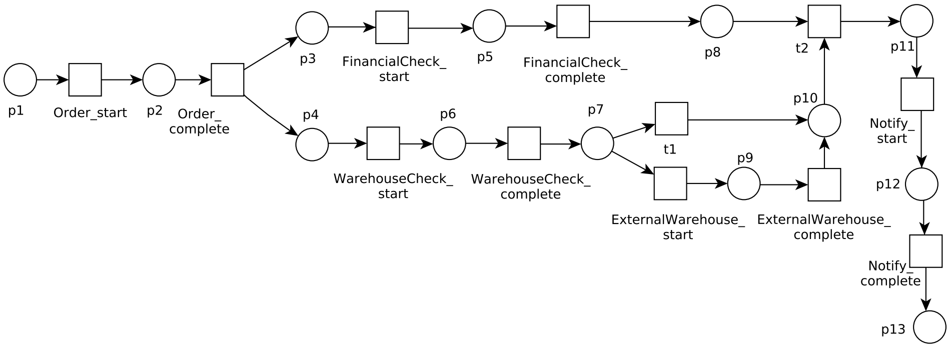


Fig. 1. Petri net for a a sale process

the availability of merchandise requested by the order is checked and, in case of a shortage, a supplying procedure is started. 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.

Note that the Petri net of Figure 1 is obtained by transforming a BPMN model using the algorithm of [5], 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 e.g. the *Join* or *Fork* gateway [5].

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. Action and timestamp are denoted respectively by $\alpha(e)$ and $\phi(e)$. Events that belong to the same process are grouped into *traces*. Formally, a trace T is a finite sequence of events $T[1], ..., T[n]$, such that $\phi(T[i]) \leq \phi(T[i+1])$ for all $i \in [1, n)$. 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.

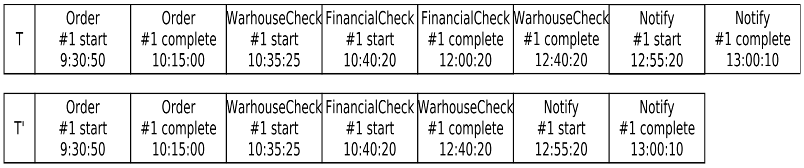
3.1 Log replay algorithm

The key algorithm exploited to analyze a Petri net model with respect to the log is the *log replay* algorithm [12][10][6]. 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. The output of a log replay execution on a trace can be represented as an ordered list R of pairs (tr, i), representing that the transition tr has been fired to mimic the event $T[i]$.

The log replay result can be used to evaluate both the conformance and performance of a business process.

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)*. Figures 2 and 3 present 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 $\{p13 \to 1\}$ and without reporting any missing tokens.

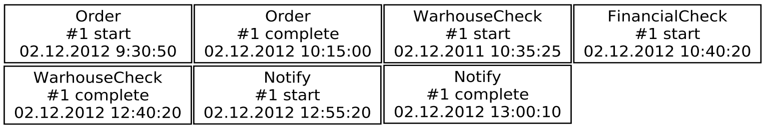


T	Order #1 start 9:30:50	Order #1 complete 10:15:00	WarhouseCheck #1 start 10:35:25	FinancialCheck #1 start 10:40:20	FinancialCheck #1 complete 12:00:20	WarhouseCheck #1 complete 12:40:20	Notify #1 start 12:55:20	Notify #1 complete 13:00:10

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. A conforme trace: T

The log replay of the trace T' terminates with remaining tokens $\{p5 \to 1, p13 \to 1\}$ and one missing token $\{p8 \to 1\}$. 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.



Order #1 start 02.12.2012 9:30:50	Order #1 complete 02.12.2012 10:15:00	WarhouseCheck #1 start 02.12.2011 10:35:25	FinancialCheck #1 start 02.12.2012 10:40:20
WarhouseCheck #1 complete 02.12.2012 12:40:20	Notify #1 start 02.12.2012 12:55:20	Notify #1 complete 02.12.2012 13:00:10	

Fig. 3. Non-conforme trace: T'.

3.3 Performance analysis

A performance analysis of a process can be performed by exploiting the log replay results. Since the logs contain timestamps, the log replay can be used to compute performance measures of the process. This is done based on the elapsed time between production and consumption of tokens. Performance analysis can be applied only to traces that do not generate missing tokens during the replay since such tokens cannot have time information [5]. During the log replay the following metrics can be computed for each trace and each place:

- sejour time (tsj) : the time interval between arrival and departure of tokens;
- synchronization time (tsc): the time interval between arrival of a token in the place and enabling of a transition in the post-set of the place;
- waiting time (tw): the time interval between enabling of a transition in the post-set of the place and token departure (thus $tsj = tsc + tw$).

The duration of an activity is an important performance measure as well, and it is given by the waiting time of the only place between the start and the complete transition of the activity.

It is worth noting that the synchronization time is greater than zero only for places which contain in their post-set transitions depending from other places. For the Petri net in Figure 1, the only places which can have a non-null synchronization time are $p8$ and $p10$. For example, the synchronization time is greater than zero for $p8$ in the trace T in Figure 2. That means that, for this particular instance of the process, the branch with the activity related to the financial checking is faster than the branch with the warehouse activities.

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 [11], 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, while y is a special attribute called *target attribute*. The target denotes the class to which a record belongs. With this technique, 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.

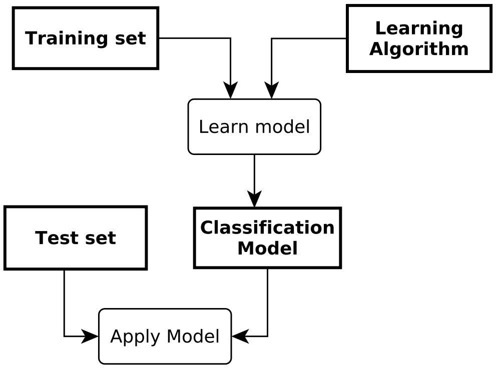


Fig. 4. General approach for solving a classification problem.

Figure 4 summarizes the general approach for solving a classification problem [9]. 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 application allows the estimation of the *model accuracy* that can be done in different ways. The simple one consists in computing the *performance* as the number of the records correctly predicted over the total number of records in the test set.

Many classification models could be used for a classification problem. For our purpose we choose the *decision tree* model since the rules detected by the classification are shown explicitly. This fact is fundamental for giving an explanation to the process behavior based on the process data values. 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, one will extract some data that will be included as a record in the dataset. For the process presented in Section 2, a record is characterized by the following attribute set: 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. Note that, since the goal of this paper is just to illustrate an approach for exploiting process data in the process analysis, the data presented here is just a synthetic data that was generated with some noise to emulate a real situation.

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.

Through existing data mining tools for classification, it is possible to build a decision tree as a classification model for the sale process example. Starting from a given event log, the attributes characterizing the process are extracted and all the resources needed for the classification algorithm employed are produced. The resulting decision tree is presented in Figure 5.

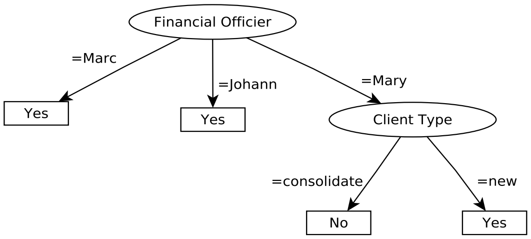


Fig. 5. Decision tree for the conformance analysis.

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, it is useful to relate what is noticed by the decision tree with the log replay results. Figure 6 presents a Petri net that summarizes the conformance analysis conducted on instances recorded in the event log taken in exam. Arcs are labeled with the number of activations done. Places, whenever they present some remaining or missing tokens, are labeled with the number of these tokens.

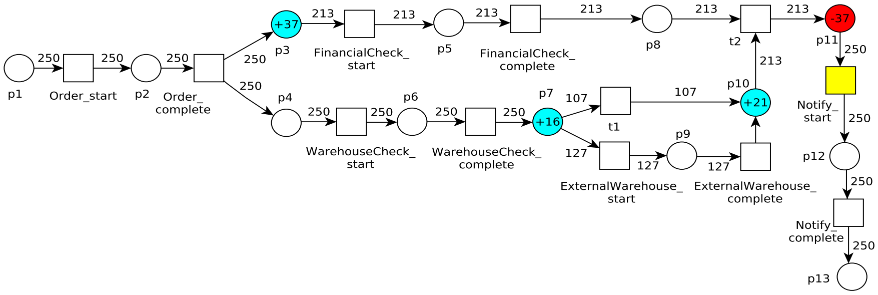


Fig. 6. Petri net: conformance results.

The Petri net in Figure 6 shows that there are 37 out of 250 process executions with conformance errors. The presence of 37 missing tokens in $p11$ signals 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 $p11$ certifies that during the replay the algorithm needs to mimic the event *Notify_start* but the transition associated to the event is not enabled in the marking reached by the net, so the algorithm creates artificial tokens for completing the trace replay. This situation is possible only if the invisible transition $t2$ is not enabled. Now, if $t2$ is not enabled that means that there are no tokens in the places $p8$ or $p10$. But notice that there are 21 remaining tokens in the place $p10$, so the transition $t2$ is not enabled for those 37 instances because the place $p8$ does not have any token. Moreover, the Petri net in Figure 6 shows 37 remaining tokens in place $p3$, so we can deduce that in those instances financial activities were not performed. From these facts, we can conclude that the conformance problems presented in 37 instances are due to a missing execution of the financial activity checking. On the other hand, the decision tree reports that the instances with conformance problems are made by consolidate clients.

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 possible and reasonable behavior in a sale procedure, consequently an extension of the business process model is needed in order to include the new scenario (*model extension activity*). Alternatively, the new scenario can be considered as an anomaly in the sales procedure. In this case, since the analysis done provides accurate information about the conformance error, corrective measures can be taken in order to avoid errors in the next orders.

The classifier constructed with this approach can be used also in a predictive way. Given a new trace recording a process execution, in order to perform the conformance checking to the process model, the decision tree can be used 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 the one needed by a decision tree to classify a new instance. Moreover, it is worth noting that whenever the set of attributes needed for the classification is known before the completion of a process execution, it can be interesting to predict the conformance result of the execution and avoid the errors that could possibly happen.

4.3 Classification for Performance Analysis

The approach presented in Section 4.2 for conformance analysis through classification can be extended to performance analysis. 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, classification can be used to 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.

OrdIde	CltIde	CltType	SalMan	FinOff	WrhsMan	Supplier	OrdResut	Synch
1	20	consolidate	Marco	Alessandro	Alex	Gianni	positive	high_wareh.
2	15	new	Anna	Mario	Alessio	Mario	positive	normal
3	10	consolidate	Maria	Roberto	Alessio	Mario	negative	normal
2	15	new	Anna	Mario	Roberto	Gianni	positive	high_wareh.
...	...	...	...	...	...	...	....	....

Table 2. Dataset for the performance analysis.

To discover how the process data may influence its performance in term of synchronization time, a classification problem is formulated for each synchronization point of the process. In this way, the model 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. For the sales process presented in Section 2, a single point of synchronization is identified. In fact, the Petri net in Figure 1 has two parallel branches, one for the financial activity and the other for the warehouse activities. We are interested in analyzing the synchronization between these branches. Indeed, the synchronization time of interest is measured in correspondence to the places $p(8)$ and $p(10)$: $tsc(p8)$, $tsc(p10)$ which are computed by the log replay algorithm as explained in Section 3.

To formulate the classification problem the dataset is extracted from the event log, whereas the target attribute is one of the log replay performance results. The target must be something that characterizes the performance in term of synchronization. An idea can be to simply consider the difference between synchronization time of $p8$ and $p10$: $tsc(p8) - tsc(p10)$. It should be noted that classification deals with discrete or categorical target attribute. Since the performance metrics, the synchronization time in this case, are expressed as continuous values, a discretization phase is required during the formulation of the classification problem. Let us choose the attribute target as a categorical attribute: given a threshold S, if $|tsc(p8) - tsc(p10)|$ exceeds the value S, then the target attribute indicates the branch with the high synchronization time, otherwise the target indicates that the instance presents a normal synchronization time. Therefore, the target attribute can assume three possible values: *high_financial, high_warehouse* and *normal*. Based on the process data extracted from the event log and target attribute computed by a discretization of the log replay results, the dataset for the classification problem is built and presented in table 2. As done for conformance analysis, the dataset is used as input of the classification algorithm that returns the decision tree presented in Figure 7.

The decision tree shows that the synchronization time is too high, so not acceptable, for those instances in which the external warehousing is done by the supplier *Gianni*. In other terms, the rule detected by the decision tree correlates the process performance, measured as the difference of two synchronization

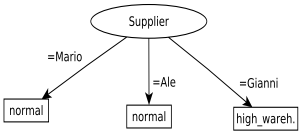


Fig. 7. Decision tree for the performance analysis.

times with the external warehousing activities. In this case, the analysis through classification identified such activities as a bottleneck in the *specific case* of the supplier Gianni. This result can be taken as a start point for more investigations that can be done in a *targeted manner* thanks to the pattern identified by the classification analysis.

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* [9], an open-source framework implementing Process Mining tools, and *Weka* [3], 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 of some *feature selection* techniques before using it as a training set for the classification task.
- `Generate Instances With Performance`: the plug-in takes as input an event log, an indication of the synchronization point to examine and the performance result computed by the log replay. The output is a training set as presented in table 2.
- `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 classification model generation and utilization for the prediction purpose:

- `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.

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.