On the interpretation of traces of low level events in business process logs

Extended abstract

Bettina Fazzinga[1], Sergio Flesca[2], Filippo Furfaro[2], Elio Masciari[3], and Luigi Pontieri[3]

[1] ICAR-CNR, Italy {fazzinga,masciari,pontieri}@icar.cnr.it
[2] DIMES, University of Calabria, Italy {flesca,furfaro}@dimes.unical.it

Abstract. We consider the scenario where the executions of different business processes are traced into a log, whose traces describe the process instances as sequences of low-level events (representing basic kinds of operations). In this context, we address the following problem: given a description of the processes' behaviors in terms of high-level activities (instead of low-level events), and in the presence of uncertainty in the mapping between events and activities, find all the interpretations of each trace Φ, in terms of the process model that Φ conforms to and of the sequence of activities that may have triggered the events in Φ. We describe the probabilistic framework presented in [7] supporting the extraction of a compact representation of Φ's interpretations, where each interpretation is associated with a probability score representing the probability of being the actual one.

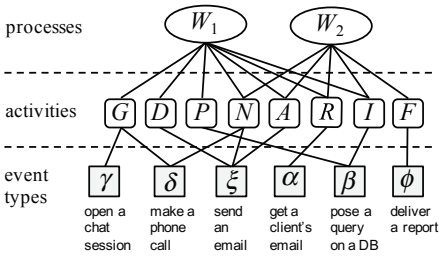
Keywords: Business processes · Graph structure · Probabilistic conditioning.

1 Introduction

Thanks to the increasing diffusion of automated tracing systems, the analysis of log data describing executions of business processes has gained momentum with the growth of the Process Mining research field, which addresses the "*confrontation between event data and process models*" [1] through new process-aware data analysis tasks, such as: inducing a process model [2], quantifying "how much" a log and a model conform one to the other [14], and supporting advanced query-based process analytics [13, 6, 8, 9].

However, all the approaches and tools developed in this field require that each log event can be mapped to well-defined activities, corresponding to some high-level view of the process. As a matter of fact, this assumption often does not hold in practice: in the logs of many processes, the events just represent low-level operations, with no clear reference to the business activities that were carried out through these operations, as shown in the following example.

SEBD 2018, June 24-27, 2018, Castellaneta Marina, Italy.



activity label	*activity description*
A	Send an alert to managers
D	Dispatch a contract proposal
F	Fix the issue
G	Get more information from the client
I	Retrieve data from the IS
N	Notify the request's outcome
P	Define a service package
R	Receive a request

Fig. 1. Running example: processes, activities and events (left), and activity descriptions (right). The graph on the left summarizes the possible mappings between activities and events, and process-activity relationships.

Example 1 (Running example). Consider the case of a phone company, where two business processes are carried out: a process W_1 for the activation of services, and an issue-management process W_2.

An abstract description of the behaviors of these processes is available in the form of loose process models, that specify which high-level activities compose each process, and several temporal constraints over the execution of the activities. The activities of processes W_1 and W_2 are shown in Figure 1 via process-activity links. In addition, assume that the following constraints are known to be satisfied by the two processes' instances: (i) every instance of either process starts and ends with an execution of activities R and N, respectively, and these activities cannot be executed multiple times in the instance; (ii) in every instance of W_1, any execution of activity P (resp. D) must be followed (resp. cannot be followed) by an execution of activity D (resp. P).

All these activities are performed by executing low-level operations (e.g., email exchanges, phone calls, database accesses). Figure 1 reports all the operations, regarded each as an event and denoted with a Greek letter, as well as their mapping to the high-level activities.

Notably, the event-activity mapping is many to many*: for example, activity G can produce an instance of either event δ or event γ, while an instance of event δ can be generated by the execution of either activity G or N. Thus, any execution of a process activity generates an instance of one of the events associated with the activity, and the sequence of all the event instances that are produced for an entire process instance is stored in the log as the* trace *of the process instance. For example, an instance of W_1 corresponding to the activity sequence $R\,I\,G\,P\,N$*

might be stored in the log in the form of a trace $\overline{\alpha}\ \overline{\beta}\ \overline{\gamma}\ \overline{\beta}\ \overline{\xi}$ *or of a trace* $\overline{\alpha}\ \overline{\beta}\ \overline{\delta}\ \overline{\beta}\ \overline{\delta}$*, where* $\overline{\alpha}$ *(resp.* $\overline{\beta}$*,* $\overline{\gamma}$*, etc.) denotes an instance of event* α *(resp.* β*,* γ*, etc.).*

Assume that we want to interpret a trace $\Phi_{ex} = \overline{\alpha}\ \overline{\beta}\ \overline{\xi}\ \overline{\xi}$ *as an execution of either* W_1 *or* W_2*, that means replacing each event in* Φ_{ex} *with one of the activities that could have generated it, while ensuring that the resulting activity sequence complies with one of the process models. Only three of such activity sequences exist for* Φ_{ex}*:* $R\ I\ D\ N$*,* $R\ I\ A\ N$*, and* $R\ P\ D\ N$*. Indeed, the sequence* $R\ P\ A\ N$ *too complies with the event-activity mapping, but it is not a valid interpretation, since it does not conform to either* W_1 *(no occurrence of* D *appears in the sequence after that of* P*) or* W_2 *(*P *is not an activity of* W_2*).*

In the process management scenario like that illustrated above, we address the following problem, called *interpretation problem*: Given a log L, containing traces generated by an arbitrary number of business processes, and a set $\mathcal{W}$ of (partial) behavioral models for the processes, we want to interpret each trace $\Phi = e_1 \dots e_m$ in L by establishing: 1) the process whose execution generated the sequence of events stored in Φ, and 2) for each low-level event e_i in Φ, which activity generated e_i as a result of its execution.

The need of bridging some abstraction gap between log events and process activities has been addressed by several recent works [11, 4, 3, 15, 12], However, these approaches cannot actually solve the interpretation problem stated above as: *(i)* they assume that all the given traces come from a single business process; and *(ii)* they return just one "optimal" interpretation for each trace. In fact, the problem stated here has been only considered in [5] and in [10] in simplified settings. Both these interpretation techniques have been considered as terms of comparison in the experimental evaluation of our approach performed in [7].

2 Our framework

In our context, an instance w of a *process* W is the execution of a sequence $A_1 \dots A_m$ of *activities*; in turn, the execution of each activity A_i yields an instance e_i of an event E_i; hence, the trace describing w consists in the sequence $e_1 \dots e_m$ of event instances. For any event e_i occurring in a trace, we assume that the starting time of its execution is stored in the log, and denote it as $e_i.t_s$. In the following, we assume given the sets $\mathcal{W}$, $\mathcal{A}$, $\mathcal{E}$ of (types of) processes, activities, and events, respectively.

We assume that every process $W \in \mathcal{W}$ is "regulated" by a "*composition rule*", that restricts the sequences of activities that are allowed to be executed when W is enacted. The set of composition rules of the processes in $\mathcal{W}$ will be denoted as $\mathcal{CR}$. More specifically, the composition rule associated with W is a tuple $\langle$*ActSet*, *StartAct*, *EndAct*, $\mathcal{IC}\rangle$, where:

- *ActSet* is the set of activities that are allowed to be executed in any instance of W;
- *StartAct* (resp. *EndAct*) $\subseteq$*ActSet* is the set of the starting (resp. ending) activities of W, which are the activities that are allowed to be executed at the beginning (resp. end) of any instance of W; and

- $\mathcal{IC}$ is a set of constraints of the form $A \Rightarrow_{\mathcal{T}} B$ (called *must-constraint*) or $A \Rightarrow_{\mathcal{T}} \neg B$ (called *not-constraint*), where A, $B \in \mathcal{A}$ and $\mathcal{T}$ is of the form '$\leq c$', where c is a constant.

Here, $A \Rightarrow_{\mathcal{T}} B$ (resp. $A \Rightarrow_{\mathcal{T}} \neg B$), that basically imposes that, within every instance of W, the beginning of an instance a of A always (resp. never) precedes the beginning of an instance b of B such that the width of the interval between the starting times of a and b satisfies $\mathcal{T}$. Omitting $\mathcal{T}$ is the same as specifying $\mathcal{T}$ ='$\leq \infty$'.

Example 2 (Running example (continued)). Continuing Example 1, it holds: $W_1.ActSet = \{G, D, P, N, A, R, I\}$, $W_2.ActSet = \{N, A, R, I, F\}$, $W_1.StartAct = W_2.StartAct = \{R\}$, $W_1.EndAct = W_2.EndAct = \{N\}$, $W_1.\mathcal{IC} = \{P \Rightarrow D, D \Rightarrow \neg P\}$, and $W_2.\mathcal{IC} = \emptyset$.

The solution of an instance $\Phi = e_1 \dots e_m$ of the interpretation problem is called *interpretation for Φ consistent with $\mathcal{CR}$* (or, simply, *consistent* or *valid* interpretation, when Φ and $\mathcal{CR}$ are understood), and is a pair $\langle \sigma, W \rangle$, where:

- σ is called *sequence-interpretation* and is a sequence $A_1 \dots A_m$ of activities, where $\forall j \in [1..m]$ $A_j \in$ *cand-act*(e_j) (where *cand-act*(e_j) is the set of activities whose execution is known to possibly generate an instance of e_j), meaning that each e_j is interpreted as the result of executing activity A_j;
- W is called *process-interpretation* and denotes a process, meaning that Φ is interpreted as generated by process W;
- σ conforms to the composition rule of W.

Example 3 (Running example (continued)). Consider trace $\Phi_1 = \overline{\alpha}\overline{\beta}\overline{\gamma}\overline{\xi}$, where $\overline{\alpha}$, $\overline{\beta}$, $\overline{\gamma}$, and $\overline{\xi}$ are instances of α, β, γ and ξ, respectively. The correspondence depicted in Figure 1 between the types of the events occurring in Φ_1 and the activities is encoded by: *cand-act*$(\alpha) = \{R\}$, *cand-act*$(\beta) = \{I, P\}$, *cand-act*$(\gamma) = \{G\}$ and *cand-act*$(\xi) = \{A, D, N\}$. On the basis of this *cand-act*$(\cdot)$, Φ_1 can be interpreted as the result of executing one of the following sequences of activities: $\sigma_1 = R\,I\,G\,A$, $\sigma_2 = R\,P\,G\,A$, $\sigma_3 = R\,I\,G\,D$, $\sigma_4 = R\,P\,G\,D$, $\sigma_5 = R\,P\,G\,N$, and $\sigma_6 = R\,I\,G\,N$. By also looking at the composition rules of the process models, we have that: (i) σ_1 is inconsistent with the composition rule of W_1, as $A \notin W_1 EndAct$, and also with W_2's composition rule, since both $G \notin W_2.ActSet$ and $A \notin W_2.EndAct$; (ii) σ_2 is inconsistent with the composition rule of W_1, since it violates $P \rightarrow D$ and it is $A \notin W_1.EndAct$, and it is inconsistent with also with the composition rule of W_2, since $G, P \notin ActSet$ and $A \notin W_2.EndAct$; (iii) σ_3 is inconsistent with the composition rules of both W_1 and W_2, since $D \notin W_1 EndAct$, $D \notin W_2.EndAct$ and $G \notin W_2.ActSet$; (iv) σ_4 is inconsistent with the composition rules of both W_1 and W_2, since $D \notin W_1 EndAct$ and $D \notin W_2.EndAct$ (also $G, P \notin W_2.ActSet$); (v) σ_5 is inconsistent with the composition rule of W_1, since it violates $P \rightarrow D$, and with W_2's composition rule, since $G \notin W_2.ActSet$; and (vi) σ_6 is consistent with the composition rule of W_1, and inconsistent with W_2's composition rule, since $G \notin W_2.ActSet$. Thus, we have only one consistent interpretation for Φ_1, that is $\langle \sigma_6, W_1 \rangle$. Consider

now the trace $\Phi_2 = \overline{\alpha}\overline{\beta\xi\xi}$, which coincides with the trace Φ_{ex} of Example 1. As discussed in that example, the only consistent sequence-interpretations for Φ_2 are $\sigma_a = R\,I\,D\,N$, $\sigma_b = R\,P\,D\,N$, and $\sigma_c = R\,I\,A\,N$, leading to the following interpretations: $\langle \sigma_a, W_1 \rangle$, $\langle \sigma_b, W_1 \rangle$, $\langle \sigma_c, W_1 \rangle$ and $\langle \sigma_c, W_2 \rangle$.

As a matter of fact, an instance of the interpretation problem admitting multiple solutions. This situation is likely to happen frequently, for the following reasons: 1) the correspondence between events and activities is many to many, so that an event instance can be interpreted as the result of different activity executions; 2) the description of the process models provided by composition rules can be rather "loose", and hence there may be process instances consistent with the composition rules of different processes: that is, the execution of a sequence of activities can be interpreted as the execution of different processes.

The presence of this uncertainty suggests to address the interpretation problem in probabilistic terms: we model the correspondence between events and activities probabilistically. In this direction, we assume that the many-to-many correspondence between events and activities is modeled by a probability distribution function (pdf) $p^a(A|E)$ returning the *a-priori* probability that an instance of event E is generated by an execution of A. For instance, $p^a(A_1|E_1) = 0.75$ and $p^a(A_2|E_1) = 0.25$ means that any instance of event E_1 is generated by the execution of either A_1 or A_2, and that the former case is three times more probable than the latter. This pdf is said to be *a-priori* since it assigns probabilities to the activities, of which the event can be viewed as the low-level result, without looking at the context where the event instance happened. That is, it provides an interpretation for the events that occur in a trace without looking at the other events occurring in the same trace.

In the same spirit, we assume given a pdf $p^a(W)$ over $\mathcal{W}$, associating each $W \in \mathcal{W}$ with the *a-priori* probability that a trace in a log encodes an instance of W. For instance, given $\mathcal{W} = \{W_1, W_2\}$, $p^a(W_1) = p^a(W_2) = 0.5$ means that, in the absence of further knowledge, W_1 and W_2 are equi-probable interpretations of any trace.

Given the a-priori pdfs $p^a(A|E)$ and $p^a(W)$, we could employ a naïve way to solve the interpretation problem, that is the following: first, assume that the event instances in Φ are independent of one another; then, use the a-priori pdf $p^a(A|E)$ to probabilistically interpret each e_i in $\Phi = e_1 \dots e_m$ as an instance of an activity A_i, and the pdf $p^a(W)$ to probabilistically interpret the whole Φ as an instance of one of the processes, say W_j. As implied by the independence assumption, every sequence-interpretation $\sigma = A_1 \dots A_m$ obtained this way will be associated with the probability $p^a(\sigma) = \Pi_{i=1}^{m} p^a(A_i|E_i)$ (that is, the product of the a-priori probabilities of the event-to-activity mappings at each step), while the process-interpretation W_j will have probability $p^a(W_j)$. In turn, the interpretation $\langle \sigma, W_j \rangle$ will have probability $p^a(\langle \sigma, W_j \rangle) = p^a(\sigma) \cdot p^a(W_j)$. Unfortunately, this naïve approach is unlikely to provide reasonable results, as explained in the following example.

Example 4 (Running example (continued)). Assume that the a-priori probabilities of the activities are $p^a(R|\alpha) = \ 1$, $p^a(I|\beta) = p^a(P|\beta) = 0.5$, $p^a(G|\gamma) = 1$,

$p^a(G|\delta) = 0.4$, $p^a(N|\delta) = 0.6$, $p^a(A|\xi) = 0.2$, $p^a(D|\xi) = 0.5$, $p^a(N|\xi) = 0.3$, and $p^a(F|\phi) = 1$, and those of the the processes are $p^a(W_1) = 0.6$, $p^a(W_2) = 0.4$. Consider trace Φ_1. Every sequence-interpretation of Φ_1 is associated with a probability, implied by p^a and the independence assumption. In particular: $p^a(\sigma_1) = p^a(R|\alpha) \cdot p^a(I|\beta) \cdot p^a(G|\gamma) \cdot p^a(A|\xi) = 0.1$; $p^a(\sigma_2) = 0.1$; $p^a(\sigma_3) = p^a(\sigma_4) = 0.25$; $p^a(\sigma_5) = p^a(\sigma_6) = 0.15$. Moreover, owing to the values of $p^a(W_1)$ and $p^a(W_2)$, we can say that Φ_1 encodes an execution of W_1 with probability 0.6, and of W_2 with probability 0.4. In turn, the probabilities of any interpretation $\langle \sigma, W \rangle$ for Φ_1 (with $W \in \{W_1, W_2\}$ and $\sigma \in \{\sigma_1, \ldots, \sigma_6\}$) is $p^a(\langle \sigma, W \rangle) = p^a(\sigma) \cdot p^a(W)$. In particular: $p^a(\langle \sigma_1, W_1 \rangle) = 0.1 \cdot 0.6 = 0.06$, $p^a(\langle \sigma_1, W_2 \rangle) = 0.1 \cdot 0.4 = 0.04$, and so on. However, we know from our running example that there is only one interpretation consistent with the composition rules, that is $\langle \sigma_6, W_1 \rangle$. Moreover, the sum of the probabilities of the consistent interpretations should be 1 (or, equivalently, every invalid interpretation should have zero probability), while, in this case, the a-priori probability of the only valid interpretation $\langle \sigma_6, W_1 \rangle$ is $p^a(\sigma_6) \cdot p^a(W_1) = 0.09$, and the other interpretations (that are all invalid) have non-zero probability.

Example 4 shows that the naïve approach, relying only on the independence assumption and the a-priori pdfs for interpreting the traces, may yield wrong conclusions. In fact, to solve the interpretation problem, we resort to *probabilistic conditioning*, that is a well-known paradigm used in the general context of forcing integrity constraints over probabilistic databases where the assumption of independence between tuples is originally used. In our case, applying the conditioning paradigm means revising the a-priori pdf $p^a(\langle \sigma, W \rangle)$ over the interpretations into $p^a(\langle \sigma, W \rangle \,|\, \mathcal{CR})$. The latter returns either 0, if the interpretation $\langle \sigma, W \rangle$ is invalid, or the ratio of its a-priori probability to the sum of the a-priori probabilities of all the valid interpretations, otherwise.

The problem of conditioning a pdf is generally complex. In our scenario, we could revise $p^a(\langle \sigma, W \rangle)$ into $p^a(\langle \sigma, W \rangle \,|\, \mathcal{CR})$ by first enumerating all the interpretations of Φ, then discarding those not satisfying the composition rules, and finally revising the a-priori probabilities of the remaining ones. Unfortunately, this approach is often infeasible, as the interpretations to deal with are too many. For instance, if, for a trace of 50 event instances, two activities are on average compatible with each event of the trace, we have to consider 2^{50} sequence-interpretations.

Our main contribution is a framework for efficiently constructing a compact representation (called *cas-graph*) of the interpretations of an input trace Φ that are consistent with $\mathcal{CR}$, and of the conditioned pdf $p^a(\langle \sigma, W \rangle \,|\, \mathcal{CR})$.

2.1 Cas-graph

Figure 2 reports a simplified example of cas-graph, built over trace $\Phi_2 = \overline{\alpha}\overline{\beta\xi\xi}$ of our running example. The nodes of the cas-graph are of the form $\langle i$, *Act*, $W \rangle$, where i is the step of the trace over which the node has been built, *Act* is the activity of which e_i has been interpreted to be an execution, and W is the

process of which Φ_2 has been interpreted to be an instance. The source nodes of the cas-graph are not built over trace steps, but they are only used as the nodes from which all the paths representing interpretations of Φ_2 as instances of a same process W start. This cas-graph represents the three interpretations $\langle \sigma_a, W_1 \rangle$, $\langle \sigma_b, W_1 \rangle$, $\langle \sigma_c, W_1 \rangle$ (encoded by the three source-to-target paths originating from the source node over W_1), and the interpretation $\langle \sigma_c, W_2 \rangle$ (encoded by the source-to-target path originating from the source node over W_2). Moreover, each source node n is assigned the conditioned probability that Φ is an instance of the process in n, whereas each edge $\langle n_{i-1}, n_i \rangle$ is assigned the conditioned probability that event e_i corresponds to an execution of the activity in n_i given that the event e_{i-1} corresponds to an execution of the activity in n_{i-1}. Basically, the probabilities of the two source nodes in Fig. 2 mean that Φ_2 is an execution of W_1 (resp. W_2) with probability 90% (resp. 10%). Moreover, the probabilities assigned to the edges encode the conditioned probabilities of the interpretations: the conditioned probabilities that $\langle \sigma_a, W_1 \rangle$, $\langle \sigma_b, W_1 \rangle$, $\langle \sigma_c, W_1 \rangle$ and $\langle \sigma_c, W_2 \rangle$ actually are the pairs $\langle$ activity sequence, process $\rangle$ that generated Φ_2 can be obtained by multiplying the probabilities of the source nodes and of all the edges appearing along the paths that encode each of these interpretations (that is, 15%(= 90% × 1 × 58.3% × 28.6% × 1), 37.5%, 37.5% and 10%, respectively).

The nodes of a cas-graph are said to be *activity nodes*. Each activity node n is defined over an event e_i of Φ, and represents an interpretation of e_i as an instance of some activity A within the execution of some process W. In a cas-graph, an activity node is the successor of activity nodes encoding alternative interpretations of the previous event e_{i-1}, and its successors are activity nodes encoding alternative interpretations of the next event e_{i+1}.

The construction of the cas-graph is performed by considering the steps of Φ in order and, at each step, by progressively building the activity nodes over the current step. Specifically, constructing an activity node n over the current step requires checking whether the interpretation that it provides of the current step is consistent (according to $\mathcal{CR}$) with the interpretations of the past events encoded by the predecessors of n. Hence, in order to efficiently support this compatibility check, each activity node n also contains a compact description of what the interpretations of the previous steps impose on the interpretations of the future steps, due to the presence of $\mathcal{CR}$ (for the sake of simplicity, this

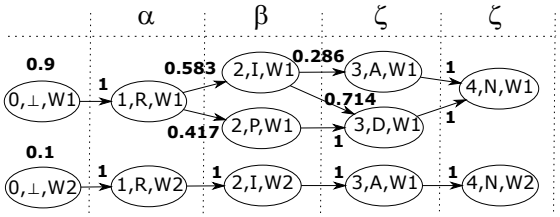


Fig. 2. A (simplified) example of cas-graph built over trace Φ_2 of our running example.

compact description was not reported in the nodes of the "simplified" cas-graph in Fig. 2). For more detail, we kindly refer the reader to [7].

3 Conclusion

We discussed the framework for efficiently interpreting business process logs composed of low-level events proposed in [7]. The proposed structure allows a compact representation of all the interpretations compatible with the constraints defined over the process models.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: Discovering process models from event logs. TKDE **16**(9), 1128–1142 (2004)
3. Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Inf. Syst. **46**, 123–139 (2014)
4. Bose, R.P.J.C., van der Aalst, W.M.P.: Abstractions in process mining: A taxonomy of patterns. In: Proc. BPM. pp. 159–175 (2009)
5. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: A probabilistic unified framework for event abstraction and process detection from log data. In: Proc. CoopIS, ODBASE, and C&TC. pp. 320–328 (2015)
6. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: A compression-based framework for the efficient analysis of business process logs. In: Proc. SSDBM (2015)
7. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: Efficiently interpreting traces of low level events in business process logs. Inf. Syst. **73**, 1–24 (2018)
8. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L., Pulice, C.: A framework supporting the analysis of process logs stored in either relational or nosql dbmss. In: Proc. ISMIS. pp. 52–58 (2015)
9. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L., Pulice, C.: How, who and when: Enhancing business process warehouses by graph based queries. In: Proc. IDEAS. pp. 242–247 (2016)
10. Fazzinga, B., Flesca, S., Furfaro, F., Pontieri, L.: Online and offline classification of traces of event logs on the basis of security risks. J. of Intell. Inf. Syst. (2017)
11. Günther, C., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In: Proc. Workshops BPM. pp. 128–139 (2009)
12. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P., Toussaint, P.J.: From low-level events to activities-a pattern-based approach. In: Proc. BPM. pp. 125–141 (2016)
13. Polyvyanyy, A., Ouyang, C., Barros, A., van der Aalst, W.M.: Process querying: Enabling business intelligence through query-based process analytics. Decision Supp. Syst. **100**, 41–56 (2017)
14. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
15. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P.: Event abstraction for process mining using supervised learning techniques. In: Proc. IntelliSys 2016: Volume 1. pp. 251–269