Extending Process Logs with Events from Supplementary Sources

Felix Mannhardt[1,2], Massimiliano de Leoni[3,1⋆], Hajo A. Reijers[1,2]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Perceptive Software, Apeldoorn, The Netherlands
[3] University of Padua, Padua, Italy
{f.mannhardt,m.d.leoni,h.a.reijers}@tue.nl

Summary. Since organizations typically use more than a single IT system, information about the execution of a process is rarely available in a single event log. More commonly, data is scattered across different locations and unlinked by common case identifiers. We present a method to extend an incomplete main event log with events from supplementary data sources, even though the latter lack references to the cases recorded in the main event log. We establish this correlation by using the control-flow, time, resource, and data perspectives of a process model, as well as alignment diagnostics. We evaluate our approach on a real-life event log and discuss the reliability of the correlation under different circumstances. Our evaluation shows that it is possible to correlate a large portion of the events by using our method.

Key words: Process Mining, Event Correlation, Data Petri Nets, Process Logs

1 Introduction

Information about the execution of processes is stored in event logs of information systems that support, monitor or enact business processes in organizations. Such event logs contain recorded sequences of events (i.e. traces). Each trace typically records the execution of a process instance (i.e. case). As organizations typically use multiple information systems, the information about a single case may be recorded across several event logs that are stored in different locations. We consider one of those logs as the main event log, for example the event log written by a central process management system (PMS), while the other event logs are seen as supplementary sources. In such a set-up, insufficient information may be available to link the supplementary events to cases in the main event log. Most existing process mining techniques require a single event log with the events grouped by cases as input [1] and, as a result, cannot make use of such unlinked data. Clearly, it would be beneficial to include supplementary events

⋆ The work of Dr. de Leoni is supported by the Eurostars - Eureka project PROMPT (E!6696) in an analysis as the events or the corresponding data may help to get a better understanding of the overall business process.

The goal of this research is to correlate unlinked events from supplementary data sources, i.e. events that have been recorded by additional systems, to the traces of the main event log. The setting for our approach is that of large organizations where various semi-autonomous units need to cooperate while using a heterogeneous set of information systems, e.g. hospitals, banks, car manufacturers. We assume that supplementary events do not contain data that allows them to be trivially associated to a single trace in the main event log. So, simple relational methods that exploit direct dependencies between data cannot be used to solve the problem. Furthermore, we assume that an expert provides domain knowledge about the entire process in form of a process model, as to identify opportunities to link the various events.

We propose an approach that uses an alignment of the process model and the main event log to correlate supplementary events to the correct traces of the main event log (cf. [2]). Fig. 1 gives an overview about the proposed technique. We use existing techniques to compute alignments for the first step (A): The alignments indicate which traces of the main event log have missing events according to the process model. We assume that those missing events may be found in a set of supplementary events by leveraging on information about the control-flow, time, resource, and data perspective of the process model as captured in the alignment. We use this information in a second step (B) to correlate supplementary events to traces in the main event log.

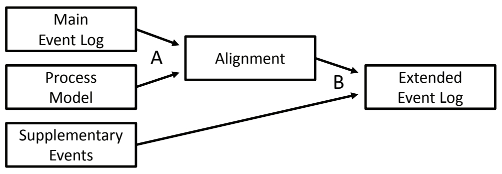

Fig. 1. Overview of the proposed approach

For example, a PMS handling credit requests may require the execution of a *financial check* activity if the *amount* of the credit exceeds €10,000. The *financial check* activity is handled by a financial system and its execution is recorded yet disconnected from the event log of the PMS. Our approach uses the constraint regarding the credit *amount* to correlate the correct events, e.g. the supplementary *financial check* events will only be correlated to traces referring to credit requests of more than €10,000.

In the literature, a few techniques have already been proposed to solve similar problems. In [3] a genetic approach, based on an artificial immune system, is used to merge multiple traces that most likely belong to the same case. While this work displays similarities to ours, it does not use an existing process model as input. As such, it cannot exploit available expert knowledge. Moreover, being a genetic approach it is likely to be slow when applied to a large data set. In fact, [3] reports an execution time of up to 10 minutes on a small data set of 8,500 traces. The approach in [4] tries to group events that belong to a case from a relational database of an ERP system by using the foreign-key relations in such databases. The work [5] describes a method to correlate events based an common attributes in the setting of web service interactions. The common assumption in [4, 5] is that there is a conceptual diagram (e.g. ER model) that can be used to link events from different sources. These approaches would not be applicable in our setting, as we explicitly assume that no such relation is defined. In [6] the input is a stream of events without case identifiers that are clustered into traces by using sequence partitioning. This technique builds only on the event name, does not make use of other data associated to the event and, therefore, only returns traces of events without data attributes. The approach presented in [7] can be used to build an event log with uniform event names out of two event logs recording executions of the same process, where events referring to the same activity have different names. This work differs from ours as traces in the first event log are not extended with events from the second event log.

This paper contributes a new technique that addresses the problem of disconnected data sources for event data. The technique employs both an alignment of observed event data with a process model and constraints defined by the control-flow, time, resource, and data perspectives of the process model to correlate events.

The remainder of the paper is organized as follows. Sect. 2 introduces preliminaries such as process models and event logs. In Sect. 3 the technique is presented. A short evaluation using a real-life event log is discussed in Sect. 4. Finally, Sect. 5 concludes with a summary and sketches future work.

2 Preliminaries

We introduce preliminaries such as *Petri net with data* (DPN-net), *Event-Log* and *Alignment* that are needed to describe our approach.

A DPN-net extends a Petri net [8] with transitions that can write variables [2]. Transitions perform *write operations* on a set of given variables and may define a data-dependent guard. Transitions can fire only if all their input places are marked and their guard is satisfied. A guard can be formulated as an expression over the process variables. We denote with $\mathit{Formulas}(X)$ the universe of such expressions defined over a set X of variables.

Definition 1 (DPN-net). *A DPN-net $N = (P, T, F, V, U, \mathit{Val}, W, G)$ is defined as:*

- *a Petri net (P, T, F);*
- *a set V of variables;*
- *a set U of variable values;*
- *a function $\mathit{Val} : V \to 2^U$ that defines the values admissible for each variable $v \in V$, i.e. $\mathit{Val}(v)$ is the domain of variable v;*
- *a write function $W : T \to 2^V$ that labels each transition with a set of* write operations*, i.e. with the set of variables whose value needs to be written/updated;*
- *a guard function $G : T \to \mathit{Formulas}(V \cup \{v' \mid v \in V\})$ that associates each transition with a different guard.*[2]

[2] If a transition t should be associated with no guard, we set $G(t) = \text{true}$.

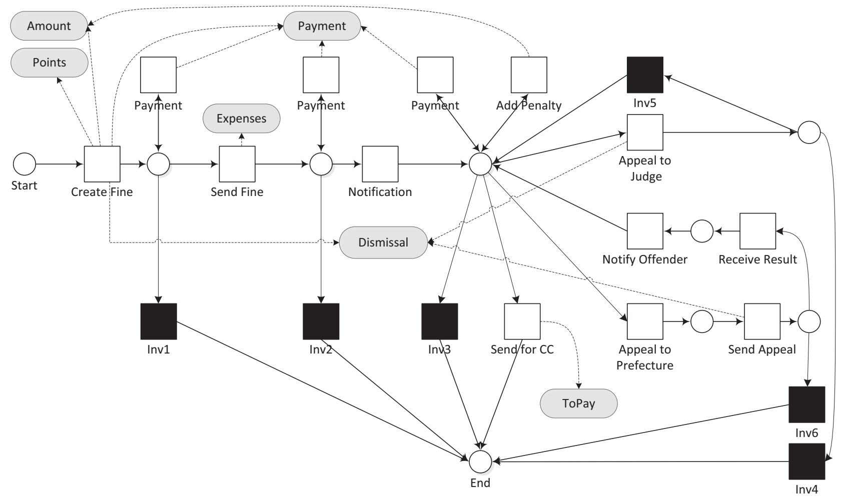

Fig. 2. DPN-net of the road traffic fine management process [2]

Table 1. Guards of the road traffic fine management DPN-net.

Transition(s)	**Guard**
Send for CC	ToPay' = Amount + Expenses - Payment ∧ ToPay' > 0
Inv2, Inv3	Payment = Amount + Expenses
Inv1	(Dismissal != NIL) ∨ (Payment = Amount ∧ Points = 0)
Receive Result, Inv5	Dismissal = NIL
Inv4	Dismissal = #
Inv6	Dismissal = G

When a variable $v \in V$ appears in a guard $G(t)$, it refers to the value just before the occurrence of t. If $v \in W(t)$, it can also appear as v' and refer to the value after the occurrence of t. Constraints on the resource and time perspective of a process can be encoded by using variables, write operations and guards of a DPN-net as shown in [2].

In the following we give an example of a DPN-net adapted from [2], which models the handling of road traffic fines by the local police in Italy.

Example 1. The DPN-net of this example is shown in Fig. 2, whereas Table 1 shows the guards associated with the DPN-net transitions. The process starts with recording the *Amount* that needs to be paid and possibly any *Points* to be deducted from the driver's license. Throughout the process the offender can pay the full fine or parts of it; the actual payment is recorded in the *Payment* variable. Whenever the fine is sent to the offender, additional postal *Expenses* need to be paid. Moreover, if the offender does not pay the fine timely, a penalty amount can be added to the fine. The offender can appeal against the fine, both through a judge or the prefecture. The result is recorded in the variable *Dismissal* and used to decide whether the process may stop. Finally, if the fine is not paid in full, the amount *ToPay* is send for credit collection (transition *Send for CC*).

Given a set X, $\mathbb{B}(X)$ denotes the set of all multi-sets over a set X. The set of possible *states* of N is formed by all pairs (M, A) where $M \in \mathbb{B}(P)$ is

Table 2. Alignment of a log trace the road fines management DPN-net

Event-Log Trace	**Process**
Create Fine {**Amount** = 36.0, **Payment** = 0.0, **Points** = 0}	Create Fine {**Amount** = 36.0, **Payment** = 0.0, **Points** = 0}
$\gg$	Send Fine {**Expenses** = 10.0}
Notification	Notification
Payment {**Payment** = 46.0}	Payment {**Payment** = 46.0}
$\gg$	Inv3

a marking of Petri net (P, T, F), i.e. a multi-set of the places in P, and A is a function that associates a value with each variable, i.e. $A : V \to U \cup \{\perp\}$, with $A(v) \in \mathit{Val}(v) \cup \{\perp\}$ for each $v \in V$. A special value $\perp$ is assigned to variables that have not been initialized.

A *transition firing* s is a pair $(t, w) \in T \times (V \not\rightarrow U)$.[3] We introduce the following functions to easily access the components of a transition firing $s = (t, w)$: $\#_{vars}(s) = w$ and $\#_{act}(s) = t$. Function $\#_{vars}$ is also overloaded such that $\#_{vars}(s, v) = w(v)$ if $v \in \mathsf{dom}(\#_{vars}(s))$, or $\#_{vars}(s, v) = \perp$ if $v \notin \mathsf{dom}(\#_{vars}(s))$. We denote the set of possible transition firings, both valid and invalid, for a DPN-net N as S_N, i.e. $S_N = T \times (V \not\rightarrow U)$. A sequence $\sigma \in S_N^*$ is called a **process trace**. Each DPN-net defines two special markings M_I, M_F: the initial and final marking. The initial state of a DPN-net is (M_I, A_I) with $A_I(v) = \perp$ for each $v \in V$. A non-empty set of *final states* exists and includes every state (M, A) with $M = M_F$. In any state, zero or more transitions of a DPN-net may be able to fire (i.e., occur). In the remainder, $\mathcal{P}_{N,M_I,M_F}$ denotes the **set of valid process traces** of a DPN-net N, i.e. the sequence of transition firings that, from the initial marking M_I, lead to final marking M_F. Readers are referred to [2] for more details.

Definition 2 (Event & Event Log). *Let $N = (P, T, F, V, U, \mathit{Val}, W, G)$ be a DPN-net and let S_N be the set of possible transition firings of N. An* event *$s_L \in S_N$ is the recording of a transition firing. A* log trace *$\sigma_L \in S_N^*$ is a sequence of events. An* event log *over N is a multi-set of traces: $L \in \mathbb{B}(S_N^*)$.*

Each event writes a value for a special variable $time \in V$ that indicates the timestamp of its occurrence; we use $\#_{time}(s_L) = \#_{vars}(s_L, time)$ to easily access the timestamp of any event s_L. We assume that L only contains events that are part of the DPN-net N. Any event referring to a transition that is not part of the process model is filtered out.

We can relate a trace σ_L of an event log L to a process model N by computing an alignment that shows where the trace deviates from the allowed behavior according to the process model. Such an alignment relates "moves" in the firing sequences of the trace to "moves" in firing sequences of the model. In case a "move" cannot be related, we explicitly denote such "no moves" by $\gg$. Table 2 shows such an alignment between a log trace and the DPN-net from Example 1.

Definition 3 (Alignment). *Let $N = (P, T, F, V, U, \mathit{Val}, W, G)$ be a DPN-net with initial marking M_I and final marking M_F. Let $S_N^{\gg} = S_N \cup \{\gg\}$. A legal*

[3] We use $\not\rightarrow$ to denote a partial function.

move in an alignment *is represented by a pair* $(s_L, s_M) \in (S_N^{\gg} \times S_N^{\gg}) \setminus \{(\gg, \gg)\}$ *such that*

- (s_L, s_M) *is a* move in log *if* $s_L \in S_N$ *and* $s_M = \gg$,
- (s_L, s_M) *is a* move in model *if* $s_L = \gg$ *and* $s_M \in S_N$,
- (s_L, s_M) *is a* move in both with correct write operations *if* $s_L \in S_N$, $s_M \in S_N$ *and* $\#_{act}(s_L) = \#_{act}(s_M)$ *and* $\forall v \in V\ \#_{vars}(s_L, v) = \#_{vars}(s_M, v)$,
- (s_L, s_M) *is a* move in both with incorrect write operations *if* $s_L \in S_N$, $s_M \in S_N$ *and* $\#_{act}(s_L) = \#_{act}(s_M)$ *and* $\exists v \in V\ \#_{vars}(s_L, v) \neq \#_{vars}(s_M, v)$,

All other moves are considered as illegal. $\mathcal{A}_N = \{(s_L, s_M) \in (S_N^{\gg} \times S_N^{\gg}) \setminus \{(\gg, \gg)\} \mid s_L = \gg \vee s_M = \gg \vee \#_{act}(s_L) = \#_{act}(s_M)\}$ *is the set of all legal moves. The* alignment *of two execution traces* $\sigma', \sigma'' \in S_N^*$ *is a sequence* $\gamma \in \mathcal{A}_N^*$ *such that, ignoring all occurrences of* $\gg$*, the projection on the first element yields* σ' *and the projection on the second yields* σ''*. An* alignment *is a* complete alignment *of a log trace* σ_L *and a DPN-net* N*, if* $\sigma' = \sigma_L$ *and* $\sigma'' \in \mathcal{P}_{N,M_I,M_F}$.

Given an alignment γ of a log trace σ_L and a process trace σ_P, $\gamma|_L = \sigma_L$ and $\gamma|_P = \sigma_P$ are referred to as the log and the process projection of γ. Each alignment is associated with a cost that accounts for deviations; the cost of an alignment is the sum of the cost of its constituent moves. The cost of moves is defined by a process analyst since it depends on the specific domain. It can be abstracted as a cost-function $\kappa : \mathcal{A}_N \rightarrow \mathbb{R}_0^+$, which associates a non-negative cost to each potential move.

Work [2] reports on a technique to compute a so-called **optimal alignment**, i.e. one of the alignments with the lowest cost. Given a log trace and a DPN-net, this technique returns a complete alignment such that any other possible complete alignment has the same or higher cost.

3 Extending Process Logs based on Alignments

This section describes our technique to extend process logs with events that are extracted from additional sources. The main input of the algorithm is *(1.)* a DPN-net $N = (P, T, F, V, U, Val, W, G)$ with a given initial and final marking, *(2.)* an event log $L \in \mathbb{B}(S_N^*)$ that records the firings of transitions belonging to some set $T' \subset T$ and *(3.)* a multiset of events $C \in \mathbb{B}(S_N)$, called *event candidates*, which records the firings of transitions belonging to some set $T \setminus T'$. Our technique aims to extend log traces in $\sigma_L \in L$ by adding events in C to σ_L in the most likely position, while guaranteeing that the alignment is still a complete one. The event candidates are assumed to not contain data that allows to trivially associate them to a trace of L (such as a trace identifier compatible with those in L).

The alignment of any log trace $\sigma_L \in L$ with N may contain a number of moves in model, each of which indicates a transition firing that was not recorded in σ_L but *should* have been observed. This missing transition firing can have two causes. The first is that it did not happen in reality at all (e.g., a process participant did not execute the expected activity deliberately or by mistake); the second reason is that it did happen but was not recorded in σ_L. Our technique elaborates a basic idea according to which a missing event (i.e. a recording of a transition firing) can possibly be found among the event candidates, if it occurred. Often, this is not as easy as it may seem. On the one hand, many event candidates may be found suitable; therefore, additional criteria need to be considered to restrict them further. On the other hand, no single candidate may be found at other times because, for example, the transition did not fire in reality. To give the reader an idea on how we restrict the number of suitable event candidates, we restrict the suitable event candidates in three steps:

1. Compute the event set $C_1 \subseteq C$ of the events that relate to the correct missing activity;
2. Compute the event set $C_2 \subseteq C_1$ that contains events that have compatible timestamps with the missing event;
3. Compute the event set $C_3 \subseteq C_2$ with events that are conforming to all data requirements of the DPN-net and, thus, are allowed to be performed in that position in the trace.

In the following we describe our technique, shown in Algorithm 1, stepwise. First, we compute an optimal alignment of each trace $\sigma_L \in L$ and N and add them to a set $\mathcal{I}$, using, e.g., the technique discussed in [2]. Variable k denotes the maximum number of event candidates that can be associated with a model move in order for a candidate to be chosen in that set. Initially, k is set to 1, which indicates that no event candidate is selected if there are potentially more than 1. The idea is that, if there are more suitable candidates, each has a lower

Algorithm 1: extendLogTraces

```
Input: DPN-net ($N$), Initial and Final Marking ($M_I, M_F$), Event Log ($L$), Event
       Candidates ($C$), Matching Limit ($l_M$)
Result: Extended Event Log ($L_C$)
$\mathcal{I} \leftarrow \bigcup_{\sigma_L \in L}$ align($N, M_I, M_F, \sigma_L$)
$k \leftarrow 1$
while $k \leq l_M$ and $C \neq \varnothing$ do
    foreach $\gamma \leftarrow \langle (s_L^1, s_M^1), \ldots, (s_L^n, s_M^n) \rangle \in \mathcal{I}$ do
        foreach $(s_L^i, s_M^i) \in \gamma$, s.t. $s_L^i = \gg$ do
            $C_1 \leftarrow \{s_C \in C \mid \#_{act}(s_C) = \#_{act}(s_M^i)\}$
            $t_E \leftarrow$ earliestTime$_N$($\gamma, s_M^i$)
            $t_L \leftarrow$ latestTime$_N$($\gamma, s_M^i$)
            $C_2 \leftarrow \{s_C \in C_1 \mid t_E < \#_{time}(s_C) < t_L\}$
            $C_3 \leftarrow \{s_C \in C_2 \mid$ replaceMove($\gamma, i, (s_C, s_C)$)$|_P \in \mathcal{P}_{N,M_I,M_F}\}$
            if $1 \leq |C_3| \leq k$ then
                $s_L \leftarrow$ selectEvent($\frac{t_E+t_L}{2}, C_3$)
                $C \leftarrow C \setminus \{s_L\}$
                $\mathcal{I} \leftarrow (\mathcal{I} \setminus \gamma) \cup$ replaceMove($\gamma, i, (s_L, s_L)$)
            end
        end
    end
    $k \leftarrow 2k$
end
return $\bigcup_{\gamma \in \mathcal{I}} \gamma|_L$
```

probability to be the right one. Therefore, we only allow to associate a candidate with a move in model if it is the only candidate. At each step of the iteration, we double this number (see line 18). This relaxation of the constraint continues until the user-defined matching limit l_M is reached or all event candidates have been matched.

As described before, the set of *matching events* for each *move in model* $(\gg, s^i_m)$ of every alignment $\gamma \in \mathcal{I}$ is refined stepwise. From line 6 to line 10, we compute the set C_1, C_2, C_3 introduced above. As mentioned after Definition 2, each event e is associated with the timestamp $\#_{time}(e)$ of its occurrence. To determine whether an event candidate should be retained for a model move $(\gg, s^i_m)$, we determine the earliest time and latest time in which transition firing s^i_m can occur (lines 7-8). The earliest time is the largest timestamp in the process projection of γ that refers to transitions that in DPN-net $N = (P, T, F, V, U, \mathit{Val}, W, G)$ comes before $\#_{act}(s^i_M)$. We call such a set[4]:

$$\begin{aligned}\texttt{preS}_N(\gamma, s) = \{s' \in \gamma|_P \mid \#_{time}(s') < \#_{time}(s) \\ \wedge\ \exists p \in {}^\bullet\#_{act}(s),\ \#_{act}(s') \in {}^\bullet p\}.\end{aligned}$$

Given a user-defined time interval b_E, the earliest time function is defined as follows:[5]

$$\texttt{earliestTime}_N(\gamma, s^i_M) = \begin{cases} \#_{time}(\texttt{first}(\gamma|_L)) - b_E & \text{if } \texttt{preS}_N(\gamma, s^i_M) = \varnothing \\ max_{s^j_M \in \texttt{preS}_N(\gamma, s^i_M)}(\#_{time}(s^j_M)) & \text{else.} \end{cases}$$

The latest time is the earliest timestamp in the process project of γ. Space limitation prevents us from discussing the latest time in detail. However, the latest timestamp refers to the smallest timestamp in the process projection of γ that refers to transitions that in DPN-net $N = (P, T, F, V, U, \mathit{Val}, W, G)$ comes after $\#_{act}(s^i_M)$.

In sum, the subset C_1 of C has been computed by considering the control-flow perspective, whereas subset C_2 of C_1 (and of C) has been determined by considering the time perspective. To reduce the set of possible candidate of C further, the data perspective is also taken into account. A candidate s_C in C_2 is also part of C_3 if we can replace the i-th move (in model) in alignment γ with (s_C, s_C)[6] and the resulting alignment is such that the process projection is still a valid process trace. This guarantees that event s_C at the i-th position conforms to the data, time and resource-perspective constraints as defined by the DPN-net N. Therefore, C_3 is the final set of event candidates.

As mentioned before, if the number of candidates is fewer than k, one is chosen. Otherwise, no candidate is chosen; possibly, one will be chosen at one of the subsequent iterations with increased k. The reason why we decide to use an incrementing k is related to fact that, first, we want to associate event candidates

[4] The preset of a transition t is the set of its input places: ${}^\bullet t = \{p \in P \mid (p, t) \in F\}$. The preset of a place p is the set of its input transitions: ${}^\bullet p = \{t \in T \mid (t, p) \in F\}$.

[5] We use $\texttt{first}(\sigma)$ to indicate the first element of the sequence σ.

[6] The replacement operation is represented in the algorithm as function $\texttt{replaceMove}(\gamma, i, \mathit{move})$ that return a variation of the alignment γ where the i-th move is replaced by *move*

with traces that have higher certainty to represent the correct matching. In this way, the event candidate set becomes smaller, which also decreases the level of uncertainty for those matching that were not sufficiently certain beforehand.

If one candidate needs to be chosen, we select the one s_L closer to the middle point between the earliest and latest time (line 12).[7] We use the middle point assuming that the waiting time between events is similar for each event. Event s_L is removed from the event candidates (line 13) and alignment is substituted by one in which the i-th move is replaced with (s_L, s_L) (line 14).

We conclude by sketching an informal discussion about the computational complexity of Algorithm 1. In the worst case, the *while* loop is repeated $\log(l_M)$ times; internally, the double *for-each* loop is repeated as many times as the overall number of model moves, which are of the same order of magnitude as the number of events in the event log. Each iteration of the double for-each loop is composed by steps that are either linear in the length of the alignments (e.g., line 7) or in the size of the event-candidate set (e.g., line 6, 9 or 10). In light of this, computing the set $\mathcal{I}$ of alignments is dominating the worst-case complexity: as discussed in [2], computing optimal alignments is, in the worst case, double exponential in the size of the log trace and number of variables and guards.

4 Evaluation

We implemented the approach as a plug-in for the open-source process mining framework ProM[8]. This section reports on the evaluation on a real-life case study about the process to manage road-traffic fines by an Italian local police. The process follows the process model discussed in Example 1. The event log contained 543,583 events grouped into 145,800 traces, out of which 76,600 are recorded process executions that are compliant with the process model.

The process is managed through a single information system and, hence, a single event log has been extracted from the transactional database underneath. To validate our technique, we assumed four scenarios:

A: Send Fine (Compliant). An independent system records the execution of transition *Send Fine*. The execution of the other process transitions is supported by a process management system. The latter generates an event log, whereas the former system only logs events without any reference to the trace to which they would belong in the event log. All events from non-compliant traces have been removed.
B: Send Fine. As above, but without removing non-compliant executions.
C: Send for CC (Compliant). The same as in A, but assuming that transition *Send for CC* was logged in a different system, instead of *Send Fine*.
D: Send for CC. As above, but without removing non-compliant executions.

[7] `selectEvent`(C, t) denotes the operation of returning the event in a set C with the closest timestamp to t.
[8] `http://www.promtools.org`

Table 3. Extending an event log in different scenarios

Scenario	|C|	l_M	% Perfectly	% Approx.	% Wrongly	% Ignored
A: Send Fine (Compliant)	33178	512	22.0	13.9	1.5	62.6
		1024	34.0	30.3	3.1	32.2
		2048	38.6	44.8	5.2	11.3
B: Send Fine	101950	512	5.1	5.1	0.4	89.5
		1024	11.6	15.5	0.8	72.1
		2048	14.1	31.5	1.7	52.8
C: Send for CC (Compliant)	28049	512	29.1	4.3	0	66.6
		1024	51.8	6.1	0	42.1
		2048	69.6	12.4	0	18.1
D: Send for CC	54232	512	20.8	3.6	3.8	71.7
		1024	29.9	4.7	5.6	59.8
		2048	40.3	6.9	7.3	45.5

We applied our technique to the four scenarios. For each scenario, we split the original event log L into an event set C and a new event log L' with the remaining events. We defined the cost function for the alignment such that a *move in model* step with events in C is cheaper than any other deviation. Then, we applied our algorithm in this way obtaining a new event log L''. Our goal is to correlate the events in C to the correct traces in L'. In the optimal case the returned event log L'' is equal to L. We evaluate the performance of our approach by using the following four types of correlations:

Perfectly correlated. Events are *perfectly correlated*, if every variable (including time) takes on the same value as in the respective event in the original trace.

Approximately correlated. Events are *approximately correlated*, if the original event refers to the same transition and the position in the extended trace is the same as in the original trace, but at least one variable has a different value from the one assigned in the original trace.

Wrongly correlated. Events are *wrongly correlated*, in case the event was not recorded at the same position in the original trace.

Ignored. All events that remain in the candidate set C and, thus, are not correlated to any trace are *ignored*.

For the evaluation, we express those criteria relative to the number of events in the *candidate set* C. If all events were *perfectly correlated*, the returned event log L'' would be indistinguishable from the original event log L. In reality, the correlation cannot be always perfect. However, for some analyzes, *approximately correlated* events are still valuable, as they are indistinguishable from the original events according to the process model. For example, the exact time that *Send for CC* was executed does not matter for some purposes. It is also possible that an event is correlated to a trace that originally did not contain such event. We do aim to minimize the amount of such *wrongly correlated* events.

The results of the experiments are shown in Table 3 for three different matching limits l_M: 512, 1024 and 2048. Looking at the *% Perfectly* column, the percentage of *perfectly correlated* events, we can see that our approach performs better on perfectly fitting event logs, but still is able to extend up to 40% of the traces in scenario D. Moreover, in scenario C, with a perfectly fitting log there are no *wrongly correlated* events in contrast to up to 7.3 % wrongly correlated events in scenario D with a non-perfect log. This could be expected: the problem of having *wrongly correlated* events tends to occur more often if deviations exists between the observed behavior and what the process model allows for, e.g. an event has not occured. For example, if an original trace in scenario D does not contain the event *Send for CC* even though the fine is unpaid, then the alignment will contain a *move in model* step for *Send for CC* and an event may get *wrongly correlated*.

It is noteworthy that the *match limit* l_M can be used to steer the reliability of the result. For lower values of l_M fewer events are *approximately* and *wrongly correlated*, at the price of an increase in *ignored* events that are not correlated. Comparing scenario A and scenario B, we noticed the fact that both the percentage of approximately and wrongly correlated events is higher in scenario A, even though only compliant traces have been considered. Moreover, in scenario B more events are ignored, i.e. 55.8 % in comparison to 11.3 % in scenario A, both for $l_M = 2048$. This observation can be explained by the larger candidate set in scenario B and the choice of the matching limit. For lower values of l_M fewer events are *approximately* and *wrongly correlated*, at the price of an increase in *ignored* events that are not correlated. Using the same matching limit, it is likely that more events are ignored for a larger candidate set and, conversely, fewer events are approximately or wrongly correlated.

Overall, the approach seems of value. If the process executions are always conformant to the model, log traces are extended with the events referring to the right transition firings (i.e. events) 80% of the time, although the variable assignments are not always perfect (i.e. approximate correlations). If the process executions are not always conformant, the accuracy of the matching degrades yet roughly 50% of the event candidates are perfectly or approximately correlated.

Please note that matching the right events is not straightforward for this event log. For instance, if a trace is extended with an event at the time 9/9/2014 instead of the correct time 8/9/2014, or with an amount of €81 instead of €80, we would already consider it as approximate matching, although from a domain viewpoint the difference could be negligible. Especially small time deviations happen often for this particular event log, as there are hundreds of traces concerning the same type of infringement, thus, requiring the same amount to be paid by the offender. There is not enough information in the event log to distinguish between any two of these traces, if they are being executed within the same time frame.

Regarding the execution time to extend the event logs with the set of event candidates, each scenario did not take more than 10 minutes, which we believe to be reasonable given the large sizes of the candidate sets, the number of traces, and the complexity of the approach.

5 Conclusion

In this paper we proposed a novel technique to extend process logs with events from supplementary sources that cannot be trivially linked to specific traces in a main event log. The problem of disconnected event sources is relevant in practice, as organizations typically use multiple information systems, which record events in separate event logs. We use an alignment of a process model and a main event log so that we can leverage on the control-flow, data, and time perspective of the process model to correlate events to a specific case. Our approach ensures that the supplementary events are correlated to traces of the main event log without violating the constraints on the different perspectives. We assume that supplementary events do not contain data that allows to easily associated them to a trace of the main event log.

A few approaches have been proposed in the literature to solve similar problems, but to the best of our knowledge no other research work makes use of the diverse perspectives of a user-supplied process model. A prototype of the technique has been realized in the ProM framework as *LogEnhancement* package. The evaluation shows promising results for a challenging real-life event log.

We acknowledge that our current evaluation is far from complete. For instance, we have only limited ourselves to remove events referring to one single transition type, such as *Send Fine* or *Send for CC*. It would be interesting to evaluate the correlation accuracy when an increasing number of transition types is removed. Similarly, we also aim to verify the approach in more real-life cases to reduce the subjectivity of the results. We aim to have a more thorough evaluation in the near future. For example, we would like to directly compare our technique with the approach proposed in [3]. However, our preliminary evaluation shows that the approach seems to be relevant in real business cases, such as the road-traffic process, which we used for our evaluation. Finally, we want to build on estimations of activity durations and use it to further restrict the possible event candidates based on timestamps.

References

1. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Mannhardt, F., de Leoni, M., Reijers, H.A. van der Aalst, W.M.P.: Balanced Multi-Perspective Checking of Process Conformance. Technical report, BPM Center Report BPM-14-07, BPMcenter.org (2014)
3. Claes, J., Poels, G.: Merging computer log files for process mining: An artificial immune system technique. In: Business Process Management Workshops. Volume 99 of LNBIP. Springer (2012) 99–110
4. Nooijen, E.H., Dongen, B.F., Fahland, D.: Automatic discovery of data-centric and artifact-centric processes. In: Business Process Management Workshops. Volume 132 of LNBIP. Springer (2013) 316–327
5. Motahari-Nezhad, H., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. The VLDB Journal **20**(3) (2011) 417–444
6. Walicki, M., Ferreira, D.R.: Sequence partitioning for process mining with unlabeled event logs. Data & Knowledge Engineering **70**(10) (2011) 821 – 841
7. Zhu, X., Song, S., Wang, J., Yu, P.S., Sun, J.: Matching heterogeneous events with patterns. In: Proceedings of the 2014 30th IEEE International Conference on Data Engineering, ICDE 2014, IEEE (2014) 376–387
8. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge University Press (1995)