Matching of Events and Activities - An Approach Based on Constraint Satisfaction

Thomas Baier, Andreas Rogge-Solti, Mathias Weske, Jan Mendling

Matching of Events and Activities - An Approach Based on Constraint Satisfaction

Thomas Baier[1], Andreas Rogge-Solti[2], Mathias Weske[1], and Jan Mendling[2]

[1] Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, D-14482 Potsdam, Germany
firstname.lastname@hpi.uni-potsdam.de
[2] Wirtschaftsuniversität Wien, Welthandelsplatz 1, 1020 Vienna, Austria
firstname.lastname@wu.ac.at

Abstract. Nowadays, business processes are increasingly supported by IT systems that produce massive amounts of event data during the execution of a process. This event data can be used to analyze the process using process mining techniques to discover the real process, measure conformance to a given process model, or to enhance existing models with performance information. While it is essential to map the produced events to activities of a given process model for conformance analysis and process model annotation, it is also an important step for the straightforward interpretation of process discovery results. In order to accomplish this mapping with minimal manual effort, we developed a semi-automatic approach that maps events to activities by transforming the mapping problem into the optimization of a constraint satisfaction problem. The approach uses log-replay techniques and has been evaluated using a real process collection from the financial services and telecommunication domains. The evaluation results demonstrate the robustness of the approach towards non-conformant execution and that the technique is able to efficiently reduce the number of possible mappings.

Key words: Process Mining, Event Mapping, Business Process Intelligence, Constraint Satisfaction

1 Introduction

Organizations conduct business processes with the support of IT systems that typically log each step made by process participants or systems in the process. Individual entries in such logs represent the execution of services, the submission of a form, or other related tasks that in combination realize a business process. In order to improve business processes and to align IT process execution with existing business goals, a clear understanding of how processes are executed is necessary. Using the event data logged by IT systems, process mining techniques help organizations to get a better understanding of their processes by discovering and enhancing process models or by checking the conformance of the execution to the specification [1]. Yet, conformance checking and enhancement of process models have one important requirement: the mapping of log entries produced by

IT systems to the corresponding process activities in the process models has to be known. Furthermore, such a mapping is not only necessary for conformance checking and process model enhancement, but it is also very helpful for discovery. The benefit of a discovered process model can only be fully exploited if the presented results use the same terminology that is known to the business analysts. Yet, such a mapping is often not existing as the logging mechanism of IT systems are usually not designed to log events for defined activities of a process model. In fact, it is often a tedious task to reconstruct a mapping from database column entries with cryptic names to the corresponding activities in the process models.

In this paper, we offer means to help the analyst to identify the mapping between a process model and events in an event log produced by an information system. Defining such a mapping is generally hard to do manually due to its combinatorial complexity. While there exist automatic techniques such as [2], they do not achieve precision and recall that would allow an analyst to accept the mapping proposal without double checking. Against this background, our contribution is a technique based on constraint satisfaction that drastically reduces the set of permissible mappings, which can be then efficiently inspected by the analyst. In contrast to recent approaches towards N:M mappings, it builds on the observation that events that are more fine-granular than model activities can be pre-processed with clustering, selection, and correlation [3, 4], which makes the identification effectively a 1:1-match problem.

The remainder of this paper is structured as follows. Section 2 states the formal concepts and gives a formal definition of the mapping problem. Having laid the foundations, the matching technique is introduced in Section 3. In Section 4 the proposed approach is evaluated using an industry process model collection and simulated event logs. Related work is discussed in Section 5 and Section 6 concludes the work.

2 Preliminaries

Let S be a finite set of states, and A be a set of activities. A process model $M = (S, s_I, s_F, A, T)$ is a transition system that defines the allowed sequences of activity executions in a business process. Here, $T \in (S \times A \times S)$ is a finite set of transition relations modeling the allowed activities in a given state that result in a succeeding state. For example $(s_1, a, s_2) \in T$ implies that we can perform activity a in state s_1 and reach state s_2. A model has an initial state $s_I \in S$ and a final state $s_F \in S$.
The function $\tau : M \rightarrow \mathcal{P}(A^*)$ captures all execution sequences that start with the initial state s_I and end in the final state s_F and are allowed in T. Note that the number of these execution sequences is infinite if the model contains loops. For example the model $M = (\{s_1, s_2, s_3\}, s_1, s_3, \{a, b, c\}, \{(s_1, a, s_2), (s_2, b, s_2), (s_2, c, s_3)\})$ has the execution sequences $\tau(M) = \{\langle a, c\rangle, \langle a, b, c\rangle, \langle a, b, b, c\rangle, \ldots\}$.

An execution sequence is also referred to as a process instance. Thus, we will use the terms execution sequence and process instance synonymously in this paper.

An IT system that supports process executions typically records events for each process instance in an event log [1]. Note that the relation of event instances to process instances might not be trivial in every practical setting. Yet, there is plenty of work on event correlation that tries to relate event instances to process instances (see e.g. [5, 6]). In this work, we therefore assume that this relation is already given. Each process instance is represented as a sequence of events $\langle e_1, \ldots, e_n \rangle$, $e_i \in E$ and also referred to as a trace θ, where E denotes the set of all events. A labeling function $\alpha : E \to \Sigma$ assigns each event a label from the set of labels Σ. In this paper we denote traces as sequences of their labels, e.g. $\langle k, l, k, m \rangle$ is a trace with four consecutive events e_1, e_2, e_3, e_4 with $\alpha(e_1) = k$, $\alpha(e_2) = l$, $\alpha(e_3) = k$, and $\alpha(e_4) = m$. An event log L is a multiset of traces.

Confronted with a process model M and an event log L, the challenge is to derive the relation between the activities $a \in A$ and the event classes $e \in E$. In this paper, we assume a 1:1 relation. Thus, we are looking for the bijective function $\mu : \Sigma \to A$ that maps event labels to their corresponding activities.

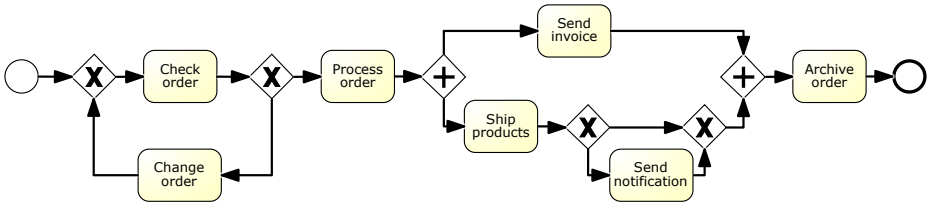


Fig. 1: Order process model in BPMN.

Having laid the formal foundations, let us look at an example. Fig. 1 introduces a simple order process that will be used to illustrate the main concepts in Section 3. Table 1 shows an exemplary event log with 5 traces that have been produced by an IT system supporting the order process depicted in Fig. 1. Obviously, it is not straightforward to interpret the given event log as the event labels are cryptic database field names that cannot be easily matched to the names of the activities in the process model. Yet, once the mapping is established as shown in Table 2, we can use the event log to check conformance between the model and the log. For example, we are able to detect that there is a case in the log, in which the order has been changed after it has already been processed. It is critical for organizations to detect, and accordingly react to such non-conformant behavior [1]. Moreover, using process discovery techniques, a new process model that reflects the actual as-is process, including all deviations, can be automatically created. If the event log contains additional data, such as timestamps, even

more techniques such as the prediction of remaining execution time for running instances [7] become possible.

Table 1: Event log (L) of order process (M).

	Label sequence
θ_1	O_CHK, O_PRC, I_SM, P_SP, O_ARC
θ_2	O_CHK, O_RCO, O_CHK, O_PRC, P_SP, P_NOT, I_SM O_ARC
θ_3	O_CHK, O_PRC, O_RCO, P_SP, P_NOT, I_SM, O_ARC
θ_4	O_CHK, O_PRC, I_SM, P_SP, P_NOT, O_ARC
θ_5	O_CHK, O_PRC, P_SP, I_SM, P_NOT, O_ARC

Table 2: Mapping μ.

Activity	**Event**
Check order	O_CHK
Change order	O_RCO
Process order	O_PRC
Send invoice	I_SM
Ship Products	P_SP
Send notification	P_NOT
Archive order	O_ARC

3 Mapping Event Log and Process Model Using Automatic Matching

This section introduces the approach for the mapping of events to given activities in a process model. The approach consists of three phases as depicted in Fig. 2.

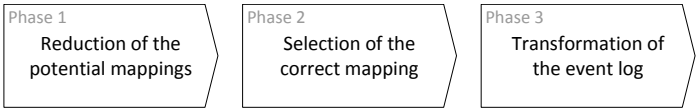


Fig. 2: Phases of the mapping approach.

The first phase is an automated phase that builds and solves a constraint satisfaction problem to reduce the number of possible mappings between activities and events. The result of this phase is a set of potential event-activity mappings. During the second phase, the analyst is guided to select the correct mapping from the derived potential mappings. Finally, the last phase is used to automatically transform one or many event logs to reflect the activities in the process model. In the following sections, we will elaborate on each of the three phases.

3.1 Reduction of the Potential Set of Event–Activity Mappings

The first phase of our approach deals with the definition of a constraint satisfaction problem (CSP) that is used to restrain the possible mappings of events and activities. A CSP is a triple $CSP = (X, D, C)$ where $X = \langle x_1, x_2, \ldots, x_n \rangle$ is

an n-tuple of variables with the corresponding domains specified in the n-tuple $D = \langle D_1, D_2, \ldots, D_n \rangle$ such that $x_i \in D_i$ [8]. $C = \langle c_1, c_2, \ldots, c_t \rangle$ is the t-tuple of constraints. We use predicate logic to express the constraints used in this paper. The set of solutions to a CSP is denoted as $S = \{S_1, S_2, \ldots, S_m\}$, where each solution $S_k = \langle s_1, s_2, \ldots, s_n \rangle$ is an n-tuple with $k \in 1..m$, $s_i \in D_i$ and every constraint in C is satisfied.

To build the CSP, first, the activities and event labels need to be mapped to the set of variables and their domains. Therefore, a bijective function $var : A \to X$ is defined, which assigns each activity to a variable with the natural numbers $1..|\Sigma|$ as domain. Moreover, a bijective function $val : \Sigma \to 1..|\Sigma|$ is defined, which enumerates event labels, i.e., which assigns each event label a natural number in the range from 1 to the number of event labels. Table 3 and Table 4 show the mapping *var* and the mapping *val* respectively for the example given in Section 2.

Table 3: Mapping *var*

Activity $a \in A$	**Variable** $var(a) \in X$
Check order	x_1
Change order	x_2
Process order	x_3
Send invoice	x_4
Ship Products	x_5
Send notification	x_6
Archive order	x_7

Table 4: Mapping *val*

Event $\alpha(e) \in \Sigma$	**Value** $val(\alpha(e)) \in 1..\|\Sigma\|$
O_CHK	1
O_RCO	2
O_PRC	3
I_SM	4
P_SP	5
P_NOT	6
O_ARC	7

Because we are looking at a 1:1 relationship between events and activities, a constraint that captures that no two activities can be mapped to the same event label can be specified. This constraint is defined as $c_1 \equiv \forall (x_i, x_j) \in X^2 : i \neq j \implies x_i \neq x_j$. It is available in most constraint solvers as the *allDifferent* constraint. With the variables, domains, and the *allDifferent* constraint defined, the solutions to the CSP reflect all possible mappings between events and activities, i.e., for n activities and events there are $n!$ solutions. For the example given in Section 2 this are be $7! = 5040$ possible mappings. In the following, we present an approach to tackle this complexity by combining the information available in the log with the knowledge of the process model structure.

To generate more constraints and thus be able to reduce the number of possible mappings, the next step of the first phase is the replay of the event log in the process model. Because the event log can contain multiple traces that encode the same ordering of events, the log is preprocessed to extract all unique variants of traces. The tuple $V = \langle v_1, v_2, \ldots, v_k \rangle$ contains all variants $v_i \in L$, $i \in 1..k$ and the tuple $W = \langle w_1, w_2, \ldots, w_k \rangle$ holds the number of occurrences for each variant, e.g., the variant v_1 is contained w_1 times in the log L. As the

example log in Table 1 only contains unique traces, v_i refers to trace θ_i, and $w_i = 1$ for any $i \in 1..k$ in the following examples.

As the relation between events and activities is unknown at this stage, the replay of one trace variant v_i is essentially a mapping of each trace variant v_i to all possible execution sequences that have the same length as v_i. Therefore, we define the relation $vpi \subseteq V \times \tau(M)$ such that $vpi = \{(v_i, pi_j) \mid v_i \in V, pi_j \in \tau(M), |v_i| = |pi_j|\}$.

The relation vpi describes possible mappings between sequences of event labels and sequences of activities. In fact, we are interested in limiting the number of possible mappings to those that result in the highest number of traces with valid execution sequences, i.e., in the mapping that yields the *maximal conformance* when replaying the log with the mapped events. Each tuple in vpi reflects a replay of a trace variant in the model. For easier explanation of the procedure, let us first assume that all traces in the log are conformant to the model. First, a constraint $c_{i,j}$ is created for each tuple $(v_i, pi_j) \in vpi$. The constraint $c_{i,j}$ reflects a mapping of event labels to activities by assigning each event label to the activity at the same position in the sequence. Hence, $c_{i,j}$ has the form $\bigwedge_{k \in 1..|v_i|} var(a_k) = val(\alpha(e_k))$. Note that there can be several paths in the model that have the same length as the trace variant. In case of conformant execution, we need to ensure that for each trace variant v_i one of these constraints holds, i.e., that one of the defined mappings allows a valid replay of the trace variant in the model. Therefore, a constraint c_i is formulated for each trace variant. The constraint c_i has the form $\bigvee c_{i,j}, \forall j : \exists (v_i, pi_j) \in vpi$.

Consider the variant $v_1 = \langle$ O_CHK, O_PRC, I_SM, P_SP, O_ARC $\rangle$. There are two execution sequences in the model that have the same length as v_1. These are $pi_1 = \langle$ Check order, Process order, Send invoice, Ship products, Archive order $\rangle$ and $pi_2 = \langle$ Check order, Process order, Ship products, Send invoice, Archive order $\rangle$. Thus, we first create the two constraints $c_{1,1} : x_1 = 1 \wedge x_3 = 3 \wedge x_4 = 4 \wedge x_5 = 5 \wedge x_7 = 7$ and $c_{1,2} : x_1 = 1 \wedge x_3 = 3 \wedge x_5 = 4 \wedge x_4 = 5 \wedge x_7 = 7$. Given the two constraints $c_{1,1}$ and $c_{1,2}$, the constraint $c_1 : c_{1,1} \vee c_{1,2}$ is derived. By adding the constraint c_1 to the CSP, we already fix the mappings of "Check order", "Ship products" and "Archive order" and thereby limit the possible mappings from $7! = 5040$ to $(7-3)! = 24$. Once the constraint c_2 for trace variant v_2 has been built in the same manner and added to the constraint satisfaction problem, the number of solutions satisfying both constraints is reduced to a single one, which is the mapping as specified in Table 2.

Yet, adding the constraint c_3 for trace variant v_3 results in a CSP that has no solution. This is due to the fact that v_3 is not compliant to the model The CSP tries to satisfy all constraints and thus requires every trace variant to be conformant. Therefore, to handle non-compliant traces, the CSP is reformulated as an optimization problem. The optimal solution to the problem is a mapping in which the maximum number of traces is conformant to the model. It is important to note that we assume that there is a sufficient number of conformant traces in the log to be able to retrieve a correct mapping.

The constraint c_i, which has been built for each trace variant, is therefore used to define a boolean variable $validVariant_i$ for each trace variant as follows:

$$validVariant_i = \begin{cases} 1 & c_i = \text{true} \\ 0 & \text{otherwise.} \end{cases}$$

The variable $validVariant_i$ reflects whether trace variant v_i represents a valid execution sequence with the chosen mapping. Having defined the variable $validVariant_i$ for each trace variant, a new variable $validTraces \in 0..|L|$ is introduced that sums up all valid traces by multiplying the valid variants with the number of traces sharing the corresponding behavior:

$$validTraces = \sum_{i=1}^{|V|} validVariant_i \cdot w_i.$$

The variable $validTraces$ is set as the optimization goal that should be maximized when solving the CSP. This way, the CSP for the example can be solved with $validVariant_1 = 1$, $validVariant_2 = 1$ and $validVariant_3 = 0$, yielding $validTraces = 2$. The optimal solution is again the correct mapping as shown in Table 2. Hence, the approach is able to deal with non-compliant traces in the log. Furthermore, this shows that it is not necessary to have a complete log containing all possible behavior in order to construct an unambiguous mapping. Note however that there are cases where it is not possible to reduce the number of solutions to a single solution. In the next section, we discuss these cases and how we can handle them.

3.2 Selection of the Correct Event–Activity Mapping

The previous section introduced the approach for an automatic matching of event labels and activities. Still, there are cases for which no unambiguous mapping can be automatically derived. Basically, this is due to two common control flow constructs: choice and concurrency. Figure 3a and Fig. 3b depict the simplest forms of these two constructs. While it is impossible to unambiguously derive a mapping for activities “A” and “B” in these two cases, it is possible for the cases depicted in Fig. 3c and Fig. 3d. This is due to the fact that for case (a) and case (b) the branches are equivalent in their behavior, while they are not in case (c) and case (d). For case (c) there are two possible trace variants. Yet, a single trace is enough to unambiguously determine a mapping between corresponding events in a log and the activities in the model, if we assume that the available event labels are known. For example, we only need a trace with the two events corresponding to activities “B” and “C”, or a trace with the event corresponding to activity “A”. Regarding case (d), there are three possible trace variants. Still, two different traces are enough to unambiguously distinguish the activities from each other, because activity “A” is the only activity that can be first and last. Note that to reach an unambiguous mapping it is required to see

at least one trace in which activity "A" was executed before, and another trace where it was executed after the parallel activities "B" and "C". If this information is not contained in the log for any reason, a certain ambiguity remains in the automatically derived mapping.

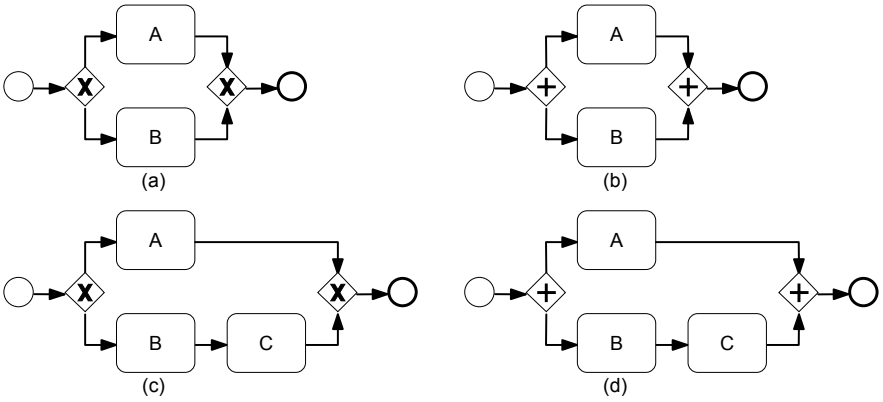


Fig. 3: Control flow patterns for choice and concurrency with different impact on potential mappings.

Summing up, there are two main sources for ambiguities in the mapping. First, choices and parallel branches with identical behavior in the branches cause ambiguities. In this case, the number of the undistinguishable branches combinatorially increases the potential mappings. The second source for ambiguities is behavior that is possible in the model but not contained in the event log.

Ambiguous mappings, i.e., cases in which the CSP has multiple solutions, cannot be automatically resolved and require a domain expert to decide the mapping for the concerned events and activities. Nonetheless, this decision can be supported by the mapping approach. To aid the analyst with the disambiguation of multiple potential mappings, we introduce a questioning approach. The analyst will be presented one event label at a time with the possible activities to which this event label can be mapped. Once the analyst decided which of the candidate activities belongs to the event label, this mapping is converted into a new constraint that is added to the CSP. Consecutively, the CSP is solved again. In case there are still multiple solutions, the analyst is asked to make another decision for a different event label. This procedure is repeated until the CSP yields a single solution. The goal is to pose as few questions to the analyst as possible. To achieve this goal, we look into all solutions and choose the event label that is assigned to the maximal number of different activities.

To illustrate this principle, consider the example trace $\theta_1 = \langle k, l, m \rangle$. The events of which should be matched to the activities in the model in Fig. 3d. Building and solving the CSP for this example leads to three solutions: $S = \{\langle x_1 = 1, x_2 = 2, x_3 = 3 \rangle, \langle x_1 = 2, x_2 = 1, x_3 = 3 \rangle, \langle x_1 = 3, x_2 = 1, x_3 = 2 \rangle\}$.

The value 2, which corresponds to event label "l" in this case, is assigned to all three variables, which correspond to the activities "A", "B" and "C". Opposed to this, the other two values are only assigned to a subset of the three variables. By deciding the matching activity for event label "l", the CSP contains only one solution. Deciding the matching for any of the other two event labels results in a CSP with two possible solutions.

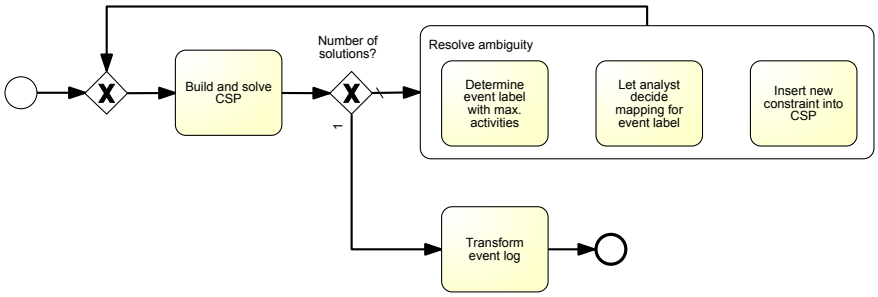


Fig. 4: Detailed flow of the matching approach.

3.3 Transformation of the Event Log

Having defined the procedure to build a CSP and iteratively resolve any ambiguities, the next step is to use the solution of the CSP to transform the event log. A single solution of the CSP can be interpreted as the mapping μ. The mapping μ can be used to iterate over all traces in the event log and replace each event label $\alpha(e_i)$ with the label returned by $\mu(\alpha(e_i))$. This results in an event log where each event carries the label of its corresponding activity. Such an event log can then be used as input for any process mining technique and other analyses of the process.

4 Evaluation

For the purpose of evaluation, the approach presented in this paper has been implemented as a plug-in in the process mining framework ProM[1]. The Petri net notation has been chosen as modeling language for the implementation of the approach, because it has well-defined semantics and can be verified for correctness [9]. Furthermore, most of the common modeling languages, as e.g. BPMN and EPC, can be transformed into Petri nets [10]. As solver for the constraint satisfaction problem, the java library CHOCO[2] has been used.

[1] See `http://processmining.org`
[2] See `http://www.emn.fr/z-info/choco-solver/`

4.1 Experimental Setup

To evaluate our approach with real life business processes, we used the *BIT process library, Release 2009* that was analyzed by Fahland et al. in [11] and is openly available to academic research[3]. The process model collection contains models of financial services, telecommunications, and other domains [11]. First, the models have been transformed into Petri nets and checked for lifelocks, deadlocks and boundedness. Models that contained lifelocks, deadlocks or where not 1-bounded have been filtered out. Furthermore, models with disconnected activities (i.e., single activities or groups of activities that are not connected to the remaining process) have also been disregarded. After the filtering step, 796 models remained with which we tested our approach. There exist some models in the collection with very large state spaces due to massive parallelism. Fahland et al. [11] report that about 8.5 percent of the models they analyzed had state spaces of over 1 000 000 states, cf. [11, Table 2]. In our case, we start with an already reduced collection, where many of the models with large state spaces were filtered out, e.g., due to unboundedness. Of the considered 796 models, only less than 3 percent led to memory problems of our algorithm. Finally, we could use 779 processes for the evaluation of our approach. For these process models, we generated event logs by simulating the processes.

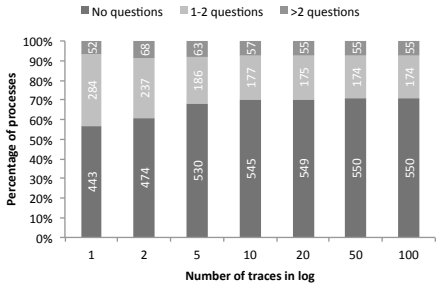
The purpose of our experiment is to evaluate (1) the *effectiveness* of our approach and (2) the *efficiency* of the method with regard to the required manual work by the process analyst. Therefore, we created two sets of event logs from the simulated processes. First, we randomly chose traces for each process to generate event logs with an increasing number of traces to show how the approach performs depending on the available number of trace variants. It is expected that the more different traces the algorithm is provided with as input, the more constraints are created on the possible mappings, which in turn make it easier for the analyst to select the correct mapping from the smaller set of resulting alternatives.

Second, to assess how the approach is able to deal with noise, we again randomly chose traces for each process, but fixed the number of traces for each process. The resulting logs were then used to create logs with different levels of noise by shuffling, duplicating and removing events for a different percentage of traces, i.e., we controlled the amount of noisy traces. For both log sets we evaluated (1) whether the approach is able to find the correct mapping, and (2) how many questions need to be answered by the analyst to arrive at the correct mapping. Moreover, we measured the runtime performance.

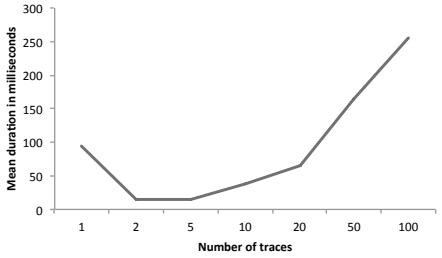
4.2 Evaluation of Effectiveness and Efficiency

Starting with the evaluation with logs of different sizes, we generated seven different event logs for each process, each event log with a specific amount of traces. Looking at (1) the *effectiveness* of our approach, it turned out that we

[3] See `http://www.zurich.ibm.com/csc/bit/downloads.html`



(a) Impact of the number of available traces on the number of necessary questions for conformant event logs.



(b) Mean duration of the mapping algorithm for different log sizes.

Fig. 5: Experiment results with varying number of traces.

were able to derive the correct mapping for all of the processes independent of the number of available traces. Figure 5a shows how many times we would have had to ask the analyst to decide to which activity an event belongs. For the set of event logs that entail only one trace, the approach is able to build the correct mapping for 443 (57 %) of the processes without any interaction with the analyst. For 284 (36 %) of the processes one or two decisions were necessary to be made manually, which is still very little effort. Only 52 (7 %) of the processes required more than two questions. The maximal number of questions for one process was six questions.

Furthermore, Fig. 5a depicts that with an increasing number of traces, the number of required manual decisions decreases until the saturation point is reached at 50 traces for all processes. Thus, it can be seen that for logs that only contain valid traces with respect to the model, our approach is both effective (i.e., it includes the correct solution even if only a small number of traces is provided as input to the algorithm) and efficient (i.e., it requires only limited interaction).

Regarding (2) the *efficiency* of the matching approach, we also measured the time that the approach takes to compute the solutions. Fig. 5b shows the mean durations for each of the evaluation sets with different log sizes. First of all, it can be stated that the approach runs conveniently fast, with a maximum average duration of 300 milliseconds. Moreover, it can be seen that there is an increase in speed depending on the trace size from one to two input traces. This increase in speed is due to the increased number of constraints, which help the constraint solver to reduce the search space. Nevertheless, the gain by the reduction of the search space is at some point outweighed with further increase of the log size as the construction of the constraints takes more time and the solver needs to evaluate a higher number of constraints that do not contribute to the reduction of the search space. Yet, the maximal duration we encountered was around 1 minute for 100 traces in the log. Hence, we argue that the approach runs fast enough in practical settings.

Turning to the evaluation with noise, we took a random set of 100 traces for each process model and randomly inserted noise into a fixed percentage of these traces. By continuously increasing the number of traces that contain noise, we evaluated the impact of noise to the effectiveness and efficiency of our approach. As result of this evaluation it can be stated that the mapping results are not influenced by the noise up to a level of more than 90 percent of traces that contain noise. Still, the increasing noise level has a negative impact on the runtime of the mapping approach. While the mapping algorithm requires on average only 350 milliseconds for logs without noise, it takes more than 10 times as long for logs in which 50 percent of the traces contain noise.

4.3 Discussion

Summing up, it can be stated that the presented approach performs well in practical settings and requires none or only very little manual interaction for the majority of cases. Yet, it has to be noted that for the case that the event log and process model are not on the same abstraction level, further work will be necessary to establish the 1:1 relationship required by the matching approach. Future research should investigate the efficient usage of clustering, correlation and filtering techniques for the establishment of a 1:1 relationship for those cases where it is not yet in place.

A limitation of the presented approach is that it cannot deal with models that contain massive parallelism. In these cases, finding all potential execution sequences through the model with the same length as the input trace is costly and cannot be handled with reasonable computational resources. Yet, more than 97 percent of the filtered models we used (i.e., those that contain no lifelocks, deadlocks, and are 1-bounded) do not have massive state spaces and can be processed in reasonable computation time. This indicates that the approach is practicable for a large share of real-life process models.

In the next section, we discuss related work and the differences to our approach.

5 Related Work

Research related to this paper can be generally subdivided into approaches working on event logs and approaches working on process models. The work that focuses on event logs can be categorized into event log abstraction and event correlation. The related techniques that work on process models fall into one of three categories: process model abstraction, process model matching and process model similarity. In both main categories—work on event logs and on process models—there are a few hybrid approaches that take both an event log and a process model as input. Yet, these approaches always focus on either log or model when it comes to the objectives and the output of those techniques.

Looking at the approaches focusing on event logs, there are several approaches aiming at the abstraction of events to activities. Günther et al. introduce in [12] an approach that clusters events to activities using a distance function based on time or sequence position. Due to performance issues with this approach, a new means of abstraction on the level of event classes is introduced by Günther et al. in [13]. These event classes are clustered globally based on co-occurrence of related terms, yielding better performance but lower accuracy. A similar approach introducing semantic relatedness, N:M relations, and context dependence is defined by Li et al. in [3]. Another approach that uses pattern recognition and machine learning techniques for abstraction is introduced by Cook et al. in [14]. Together with the fuzzy miner, Günther and van der Aalst present an approach to abstract a mined process model by removing and clustering less frequent behavior [15]. While all these approaches aim at a mapping of events to activities, they are designed to automatically construct activities and not to match events to activities that have already been defined a-priori. In [2], we introduced an approach that aims at to mapping of events to pre-defined activities. Yet, this approach still required more manual work as the precision of matchings is not sufficiently high. In contrast, the approach presented in this paper requires only very little manual effort to match events to pre-defined activities.

The second branch of related approaches working on event logs are those works that deal with event correlation, as the work by Perez et al. in [5]. The main objective of event correlation techniques is to group events belonging to the same process instance. Typically, event attributes are investigated to find similarities. These techniques are similar to our approach in the fact that they also look at a relation between a set of events and a set of other entities. Yet, these approaches specialize in finding a 1:N mapping by clustering events of different types based on similarities and are therefore not suited to address the 1:1 mapping problem between events and predefined activities. In fact, we assume that the correlation of events to process instances is either already given, or can be established by an approach like [5].

Our work is also related to automatic matching for process models. While matching has been partially addressed in various works on process similarity [16], there are only a few papers that cover this topic as their major focus. The work on the ICoP framework defines a generic approach for process model matching [17]. This framework is extended with semantic concepts and probabilistic optimization in [18, 19]. Further, general concepts from ontology matching are adopted in [20]. The implications of different abstraction levels for finding correspondences is covered in [21]. However, all these works focus on finding matches between two process models, not between events and activities.

6 Conclusion

In this paper we introduce a novel means for the mapping of events to activities that can be used as a preprocessing step to enable business process

intelligence techniques (e.g., process mining). The approach uses behavioral information stored in existing business process models and the execution order of events generated by IT systems to establish a connection between conceptual process models and operational execution data.

The approach distinguishes from current works by establishing a 1:1 relation between events and a given set of activities in a process model. As we have shown in the evaluation in Section 4, the newly introduced matching technique performs well and requires very little manual intervention. It is also robust towards noise. Yet, there are a few processes that cannot be handled due to their very large state spaces. Future work needs to investigate, how such processes can be handled efficiently.

References

1. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
2. Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Information Systems (2014)
3. Li, J., Bose, R., van der Aalst, W.M.P.: Mining context-dependent and interactive business process maps using execution patterns. In: BPM'2010 Workshops, volume 66 of LNBIP, Springer (2011) 109–121
4. Nezhad, H.R.M., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. VLDB J. **20**(3) (2011) 417–444
5. Pérez-Castillo, R., Weber, B., de Guzmán, I.G.R., Piattini, M., Pinggera, J.: Assessing event correlation in non-process-aware information systems. Software and Systems Modeling (2012) 1–23
6. Rozsnyai, S., Slominski, A., Lakshmanan, G.T.: Discovering event correlation rules for semi-structured business processes. In: Proceedings of the 5th ACM International Conference on Distributed Event-based System. (2011) 75–86
7. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic Petri nets with arbitrary firing delays. In: Service-Oriented Computing. Volume 8274 of LNCS. Springer Berlin Heidelberg (2013) 389–403
8. Freuder, E., Mackworth, A.: Constraint satisfaction: An emerging paradigm. In: Handbook of Constraint Programming. Volume 2 of Foundations of Artificial Intelligence. Elsevier (2006) 13–27
9. van der Aalst, W.M.P.: Verification of workflow nets. In: ICATPN. Volume 1248 of LNCS. Springer (1997) 407–426
10. Lohmann, N., Verbeek, E., Dijkman, R.M.: Petri net transformations for business processes - a survey. T. Petri Nets and Other Models of Concurrency **2** (2009) 46–63
11. Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Analysis on demand: Instantaneous soundness checking of industrial business process models. Data & Knowledge Engineering **70**(5) (2011) 448–466
12. Günther, C.W., van der Aalst, W.M.P.: Mining activity clusters from low-level event logs. In: BETA Working Paper Series. Volume WP 165., Eindhoven University of Technology (2006)
13. Günther, C.W., Rozinat, A., van der Aalst, W.M.P.: Activity mining by global trace segmentation. In: BPM Workshops. Volume 43 of LNBIP. (2009) 128–139

14. Cook, D.J., Krishnan, N.C., Rashidi, P.: Activity discovery and activity recognition: A new partnership. IEEE T. Cybernetics **43**(3) (2013) 820–828
15. Günther, C.W., van der Aalst, W.M.: Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. In: BPM. Volume 4714 of LNCS. Springer Berlin Heidelberg (2007) 328–343
16. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Information Systems **36**(2) (2011) 498–516
17. Weidlich, M., Dijkman, R.M., Mendling, J.: The ICoP framework: Identification of correspondences between process models. In: CAiSE 2010. Volume 6051 of LNCS., Springer (2010) 483–498
18. Leopold, H., Niepert, M., Weidlich, M., Mendling, J., Dijkman, R., Stuckenschmidt, H.: Probabilistic optimization of semantic process model matching. In: BPM. Volume 7481 of LNCS. Springer Berlin Heidelberg (2012) 319–334
19. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing recall of process model matching by improved activity label matching. In: BPM. Volume 8094 of LNCS. Springer Berlin Heidelberg (2013) 211–218
20. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer-Verlag (2007)
21. Weidlich, M., Dijkman, R., Weske, M.: Behaviour equivalence and compatibility of business process models with complex correspondences. The Computer Journal **55**(11) (2012) 1398–1418