Case and Activity Identification for Mining Process Models from Middleware

Saimir Bala[1(✉)], Jan Mendling[1(✉)], Martin Schimak[2], and Peter Queteschiner[3]

[1] Vienna University of Economics and Business, Vienna, Austria
{saimir.bala,jan.mendling}@wu.ac.at
[2] plexiti GmbH, Wien, Austria
martin.schimak@plexiti.com
[3] Phactum Softwareentwicklung GmbH, Wien, Austria
peter.queteschiner@phactum.at

Abstract. Process monitoring aims to provide transparency over operational aspects of a business process. In practice, it is a challenge that traces of business process executions span across a number of diverse systems. It is cumbersome manual engineering work to identify which attributes in unstructured event data can serve as case and activity identifiers for extracting and monitoring the business process. Approaches from literature assume that these identifiers are known a priori and data is readily available in formats like eXtensible Event Stream (XES). However, in practice this is hardly the case, specifically when event data from different sources are pooled together in event stores. In this paper, we address this research gap by inferring potential case and activity identifiers in a provenance agnostic way. More specifically, we propose a semi-automatic technique for discovering event relations that are semantically relevant for business process monitoring. The results are evaluated in an industry case study with an international telecommunication provider.

Keywords: Business process management · Process monitoring
Process mining · Case identification

1 Introduction

Business process monitoring is a key step towards improvement. In practice, Business Process Management (BPM) solutions are implemented over multiple independent systems. While system integration is a common endeavor in this scenario, monitoring techniques are only available for the single systems and not for the entire system-landscape. For instance, in the telecommunications industry, a customer order request is typically processed through various systems

This work has been funded by the Austrian Research Promotion Agency (FFG) under grant 862950 (Business Process Optimization Toolkit).

R. A. Buchmann et al. (Eds.): PoEM 2018, LNBIP 335, pp. 86–102, 2018.
https://doi.org/10.1007/978-3-030-02302-7_6

for checking its contract conditions, available credit and promotions, several consents to the treatment of data, and finally activating the contract. Therefore, in order to monitor the business process, it must be taken into account that traces span over several systems and that identifiers for cases and activities are not known a priori.

This problem has been approached in two ways, namely *(i) manual integration* and *(ii) automatic matching*. Manual integration techniques are typically ad-hoc engineering solutions that exploit domain knowledge about events to be monitored. This class of techniques focuses on specific key events and often leave out interesting patterns that happen as a consequence of other events that were not manually selected as monitoring-relevant. Instead, automatic matching techniques aim to identify relevant events and relationships assuming no prior domain knowledge. Existing literature has addressed several challenges of automatic matching. Two main techniques are *case matching* [6,18] and *mapping* [3,7] of events to activities at different abstraction layers. Case matching approaches strive to reconstruct case identifiers compatible with eXtensible Event Stream (XES). Mapping approaches either assume the presence of a case identifier or make use of domain knowledge. These techniques help getting insights on extant relationships between events. However, they shortfall in practical situations where the event schema is not known a priori and events may have many possible case identifiers.

This paper addresses the problem of monitoring the business process using event data generated from independent systems. These events are available at different levels of granularity and more than one event can correspond to a business activity. Therefore, the first step towards process monitoring is the identification of event attributes that can serve as identifiers for cases and activities. We propose a semi-automatic approach for constructing system-spanning traces from a pool of events. Input to the approach is a set of *heterogeneous* events. We assume that these events contain data that are relevant for monitoring, but no prior knowledge of the event schema. Guided by these assumptions, the approach proposes identifiers for events and relations that are relevant for process monitoring. Thus, we position our contribution as a preprocessing technique to identify potentially interesting perspectives for the analysis of event logs. In particular, this research helps companies select attributes for creating event logs that can be analyzed with process mining techniques.

The rest of the paper is structured as follows. Section 2 illustrates the problem inspired from a real world telecommunications provides scenario, and elaborates on the state of the art. Section 3 introduces our approach to identify relevant events and attributes that can serve as identifiers. Section 4 evaluates our approach against an industry use case. Section 5 concludes the paper.

2 Background

In this section we elaborate on the addressed problem and related work.

2.1 Problem Illustration and Motivation

An international telecommunication provider has different *sales channels* to its customers. A sale channel defines a way in which customers can interact with the provider. There are two types of interactions: *(i)* direct interaction, when the customer contacts the provider directly, and *(ii)* indirect interaction, when the customer contacts the provider through partners or intermediaries. In order to handle these interactions, the company has developed a solution that relies on a middleware to connect systems from the provider and its partners. Figure 1 illustrates the architecture in the Fundamental Modeling Concepts (FMC) notation[1]. The middleware operates as a bridging layer that enables communication among IT systems from the different parties. Up to a limited period of time, it is possible to access historical events by means of queries. Given the different technologies involved, these queries usually return *heterogeneous* events.

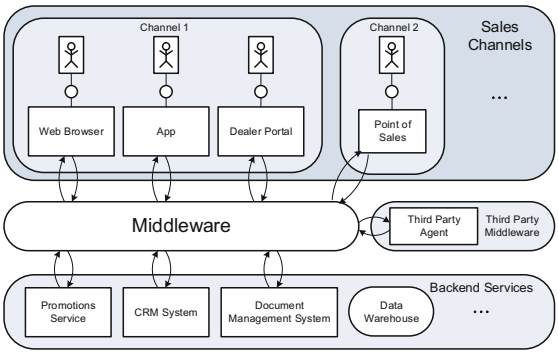


Fig. 1. Middleware bridging events from different systems

Let us illustrate the related challenges by the help of a simplified example. Figure 2 shows a typical *ordering process* from practice in the Business Process Model and Notation (BPMN). The process starts when an order is received by the order handling system and consists of two phases: *(i)* place order and *(ii)* confirm order. In the first phase an order is received from the client. A client is a software agent used by the customer, e.g., a web browser, a smartphone application or another third party system (cf. Figure 1). After a number of successful validity checks, the order is finally accepted. In the second phase, a confirmation

[1] http://www.fmc-modeling.org.

of the identity of the customer is expected. For instance, the identity confirmation can is carried out using web identification. Upon notification of successful customer identification, the order is further processed and confirmed. This concludes the ordering process.

Traces of the process are scattered across the systems that serve the different activities of the process. Because all the events are routed by the middleware, it is possible to collect these traces by querying the architecture and obtaining a set of middleware events. These middleware events are wrappers for other more fine granular events intended to reach the various systems connected. Given the different provenance of the events it is hard to have full knowledge on the database schemata of each of the source systems. Therefore, it is a challenge to understand whether an event is semantically meaningful for process monitoring.

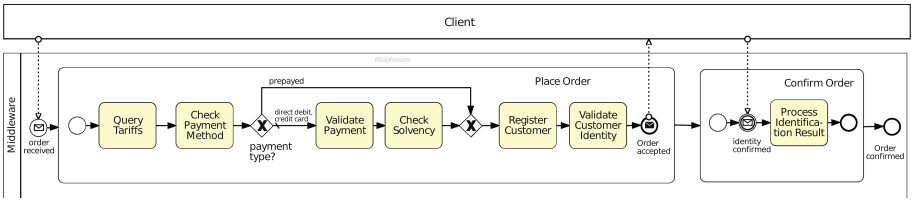


Fig. 2. A typical two step ordering process.

Table 1 shows a simplified excerpt of middleware events at a specific time interval. For illustration purposes we report only eleven attributes. In practice the number of attributes can grow up to more than forty attributes per event. In this case, we have the following: *(i) Event* is a friendly event name; *(ii) id* is a unique identifier given by the middleware; *(iii) mTrId* is transaction id representing a session in the middleware, i.e., a number of activities executed by the middleware for a specific request; *(iv) payload* carries data about the original event in different formats, e.g., JSON data from forms, SOAP XML from remote procedure calls, etc.; *(v) timestamp* reports the time the event appeared in the middleware; *(vi) sysId* may contain the name of the system that generated the event or the system that is intended to consume it; *(vii) inMethodName* contains the name of a specific method in one of the connected systems intended to consume this event; *(viii) outMethodName* contains the name of a specific method in one of the connected systems that produced this event; *(ix) uName* is an attribute that identifies the name of the user that is performing an action in the system, such as for instance requesting a new SIM card; *(x) oName* is the name of owner of the credit card requested for the payment.

As shown in Table 1, in practice events lack an explicit notion of case identifier and there might be multiple candidates that can be used. While it is true that some attributes can be checked in the documentation, it requires cumbersome coordination effort to obtain such documentation from all the partners. Moreover, in many practical settings documentation is often missing or outdated.

Although it is hard to automatically extract the business process from these data, yet, monitoring the business process is crucial. Therefore, it is useful to proceed in a schema agnostic way to support the engineer for a first classification of the attributes.

In the light of these considerations, we formulate the challenge as "**RQ:** *how to semi-automatically identify attributes that can serve as case or activity identifier*?". Note that the steps of a customer journey across the systems can be observed from different perspectives. For instance, depending on what attribute is chosen as a case identifier, it is possible to observe either the different journeys of a customer or the various customers for each journey in the systems landscape. In this sense, the problem of finding case identifiers is dual to the problem of selecting activity identifiers. Therefore, it can be assumed *orthogonality* between activities and cases in the way they partition the event space.

Table 1. An excerpt of events and their attributes managed by the middleware. The number of attributes in the real case is more than forty.

Event	**id**	**trId**	**mTrId**	**payload**	**timestamp**	
e1	By	FE7	MA1	JSON Data	2017-11-30 12:35:27.003	
e2	B0	FE7	MA2	JSON Data	2017-11-30 12:35:27.065	
e3	u8	FE7	MA3	SOAP XML	2017-11-30 12:35:27.353	
e4	vB	FE8	MA1	SOAP XML	2017-11-30 12:35:27.456	
e5	vD	FE8	MA2	SOAP XML	2017-11-30 12:35:27.488	
e6	vG	FE8	MA3	SOAP XML	2017-11-30 12:35:27.497	
e7	Os	FE9	MA1	SOAP XML	2017-11-30 12:35:27.575	
e8	Vi	FE9	MA2	SOAP XML	2017-11-30 12:35:27.575	
e9	Ox	FE9	MA3	Text	2017-11-30 12:35:27.615	
Event	**system**	**uName**	**oName**	**inMethod**	**outMethod**	**...**
e1	s1	Bob	Bob	/api/ordering		...
e2		Alice	Bob		/api2/ordering	...
e3	s2	Claire	Alice	ReqProms		...
e4	s2	Claire	Alice	ReqProms	ReqProms	...
e5	s2	Claire	Alice	CheckSolv		...
e6	s2	Claire	Alice		CheckSolv	...
e7	s3	Bob	Bob	QueryTarifs	QueryTarifs	...
e8	s3	Claire	Alice	RegCustomer		...
e9	s3	Claire	Alice	ProcIdRes	ProcIdRes	...

2.2 Related Literature

The problem has been addressed in the literature from different angles. Existing approaches can be classified into two main areas: *(i)* process mining research and *(ii)* database research. In the first category, a similar problem was tackled in [15,19]. Specifically, these two works consider the same problem setting

but they only assume two attributes, namely id and message. In this work, we assume that it is possible to distinguish different attributes, although schema information is missing. Differently from the mentioned works, we abstract case and activity indicators. This provides a step beyond the simple identification of atomic, disjunctive and conjunctive correlation-conditions. Other approaches in this area rely on the use of process mining [2] techniques to reconstruct a process model from event logs [18,20]. Because these techniques work with "flat" event logs, i.e., unaware of the granularity and multidimensionality of the events, efforts were made on how to create flat event logs from multidimensional data. This case is supported by practical scenarios where business events are stored in several tables of relational databases. Often, these data are stored by humans and therefore the log quality is low. Work from [6] tackles this problem by finding correlations and case identifiers among events. The problem of event granularity has been especially tackled in [4,5,14]. As a result a many-to-many mapping technique was defined that is able to recognize business activities from groups of fine granular events based on time distance. In further work [3], event matching is tackled my mining declarative rules from the model and the log. The problem of discovering subprocesses has been treated in [9]. Here, the authors develop a technique to discover the process model including subprocesses, instead of flat model for as is analysis. The problem is also related to multi-instantiation of sub-processes. The work in [21] tackles such a problem for process discovery and conformance checking purposes.

Efforts from the database area also have links to processes. In [7], the authors propose an approach based on describing event logs with annotations of a conceptual model of the data. The technique takes as an input *(i)* an ontology in the Web Ontology Language (OWL) language; *(ii)* an Ontology-based Data Access (OBDA) mapping specification; *(iii)* and the schema annotations specifying cases and events [8]. The output consists of an event log in the XES format. This technique helps with automatic obtaining a customized view on the process. As a drawback, knowledge about the schema is necessary.

This paper is also related to foreign-key extraction. Work from [17] tackles this problem in the context of extracting the artifact lifecycle from multidimensional events. They identify table importance based on entropy of its attributes. However, the approach does not tackle event granularity issues. The last related stream of research regard the use of Natural Language Processing (NLP) to identify process cases and activities. Contributions in this group the have focused on process discovery. In [1] a model is discovered from group stories. Work from [11] reaches 77% of accuracy in reconstructing process models from text. In [16] legacy systems code is analyzed to infer business process rules and activities. The work of [10] uses NLP to aid the extraction of artful processes from knowledge workers emails. This body of contribution indeed suggests that valuable process insights can be obtained from unstructured data. In our setting, this is particularly useful when dealing with messages or payload of events. Differently from NLP works, we work with semi-structured data, i.e. our data can (roughly) be represented in tabular format, but the schema in unknown.

In light of the described literature, we focus on the gap of semi-automatically identifying case and activity candidates.

3 Engineering Approach for Process Monitoring

Next, we present the approach for addressing the problem.

3.1 Approach Overview

We devise a three step approach to produce flat event logs. Figure 3 illustrates this approach, which takes as input a pool of heterogeneous events and proceed as follows.

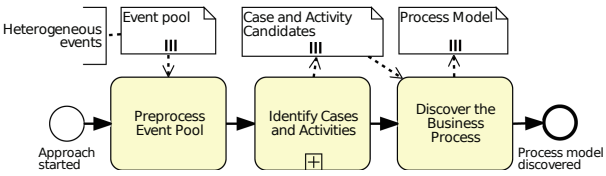


Fig. 3. Overview of the approach for extracting the business process

Step 1. Preprocess Event Pool. In this step, we extract the heterogeneous events from the middleware. Because these events contain diverse attributes, they need to be further processed and pooled together in such a way that they can be analyzed automatically. This includes enriching existing events with new attributes extracted from their payload data. A high number of events is generated in real-world scenarios. Therefore filtering techniques must be taken into account to rule out events that we do not want to monitor.

Step 2. Identify Cases and Activities. In this step, a mapping from events to activities is established. Note that business activities are not unequivocally represented in the event log. In fact, the problems of granularity and multiplicity between events and activity must be taken into account [5]. The input of this step is an enriched log with labeled cases. The output is a set of pairs that represents what can be considered as cases and activities.

Step 3. Discover the Business Process. In this step, the approach exploits the results of the previous steps to show a business process model. In particular, this step combines the case identifier, activity and timestamp for constructing an event log. The log is then converted to XES and a process mining algorithm is used for discovering the process.

3.2 Preliminaries

Next, we formalize the preliminary concepts required by our technique. We formally define heterogeneous events from multiple systems passing through the same channel as *pool of events*.

Definition 1 (Pool of Events, Attribute). Let $\mathcal{E}$ be the universe of all events. A pool of events $PL \subseteq \mathcal{E}$ is a set of recorded events in the process. Each event $e \in PL$ has attributes. Let AN be a set of attribute names. For any event $e \in PL$ and $n \in AN$, $\#_n(e)$ is the value of attribute n for event e. If an event e does not contain an attribute n, then $\#_n(e) = \perp$.

Events also contain data, which are referred to as *payload*. That is, for each event $e \in PL$, the payload is an attribute p such that $\#_p(e) \neq \perp$ where $p \in AN$, contains additional information about the event. Moreover, we assume that every event has a timestamp attribute $ts \in AN$ such that $\#_{ts}(e)$ marks the time e occurred in the middleware. Our definition of events pool does not have a notion of a case identifier. We assume that the data is recorded from different systems, and there is no unique case connected to events. Hence, the goal is to identify the most suitable case identifier among the attributes.

Every run of a process instance is a finite sequence of events, also known as trace $\sigma \in PL^*$, where PL^* is the set of arbitrary length traces. For example, $\sigma_1 = \langle e_1, e_2, e_3, e_4 \rangle$ is a trace constituted by a sequence of four events. In our problem setting there may be different ways in which events form a trace. We assume that events can be grouped into traces if there is a relation among them. Given that we have no prior knowledge on the data schema, we do not enforce the choice of any particular attribute for grouping events into traces.

Events in the middleware are produced as a result of activities that happen at business process level. An activity corresponds to the execution of a certain task that is business relevant. Examples of activities are *Query Tariffs*, *Check Payment*, *Register Customer*, and so on. We consider an N:1 mapping between the events and activities, i.e., for each business activity one or more events may occur in the middleware. Given this relation, it is possible to construct a log from activities by matching events onto activity traces. The challenge is to find a mapping $M : PL \rightarrow A$ from the pool of events to the set of activities A. That is, we aim to find a surjective function m as defined in Eq. 1 for that establishes this mapping.

$$\forall a \in A, \exists e \in PL \; : \; a = m(e) \tag{1}$$

Activities can be executed in many different orders. For process monitoring, we are interested into sequences of activities that may represent a full end-to-end execution of one process instance.

3.3 Approach

With the definitions presented above, we describe the steps for identifying case and activity attributes and extracting the business process.

Identifying Case and Activity Candidates. As discussed in Sect. 2.1, we assume that cases and activities almost partition the space of events orthogonally. Unquestionably, this assumption leaves out situations where activities appear multiple times within a case (e.g., rework loop). Nevertheless, we are interested at having a first characterization of process variants. Therefore we focus only on business processes where activities are unique within a case.

A characteristic of cases is that they uniquely identify traces (i.e., set of activities). At the same time, many activities usually belong to a case, i.e., many singular activities are labeled with the same case identifiers. Therefore, if we consider a sequence of attribute values in a trace of events $\#_\sigma = \langle \#_1, \ldots, \#_n \rangle$ then we must find at least i, j for which $\#_i = \#_j$. This condition states that case identifiers must be non unique. For an attribute A, we measure its *repetitiveness* as the fraction of unique values over the cardinality of all possible values of the attribute.

$$Rep(A) = 1 - \frac{|uniq(A)|}{|\#_A|}$$

For example, given $A = (1, 2)$ and $B = (b, b)$, then $Rep(A) = 1 - 2/2 = 0$ whereas $Rep(B) = 1 - 1/2 = 0.5$. This observation allows us to filter out attributes like the timestamp or other identifiers that are introduced by the middleware but do not represent business relevant information. For instance, timestamps present a low level of repetitiveness in a log.

Repetitiveness alone is not sufficient for determining case identifier attributes. In fact, attributes which do not have low level of repetitiveness are not necessarily case or activity candidates. We refer to these attributes as *noise*. One example is the attribute $score = None$ for all events, or attributes that only contain empty values. We can filter out these type of attributes by relying again on the almost orthogonality condition between case identifiers and activities. More specifically, to overcome erroneous inclusion of noise, we consider pairs of events which have high individual repetitiveness but a low pairwise repetitiveness, as follows.

Given a sequence of attributes, we compute the pairwise repetitiveness (a.k.a. co-repetitiveness) with Algorithm 1. The algorithm takes as input the whole set of attributes and returns the set PWR of all pairwise repetitiveness scores. Because the number of attributes in our problem setting is high, we use this measure to sort the attributes in increasing order of co-repetitiveness, namely $Sort(PWR)$. Candidate attributes A_c for case identifier are those who belong to a pair ranked on top $Sort(PWR)$ and score a high $Rep(A_c)$.

Algorithm 1. Computing pairwise repetitiveness

Input: An event pool PL, where AN is the set of its attribute names
Result: A set $PWR = \{(a_i, a_j, r)\}$ where a_i,a_j are attribute, r is their co-repetitiveness

$PWR \leftarrow 0$;
forall the e *in* PL **do**
  forall the (i, j) *such that* $i \neq j$, *with* $a_i, a_j \in AN$ **do**
    $V_i \leftarrow \#_{a_i}(e)$; `/* get all values for attribute name` a_i `*/`
    $V_j \leftarrow \#_{a_j}(e)$;
    $r \leftarrow 1 - \frac{|V_i \cap V_j|}{max(|V_i|,|V_j|)}$;
    $PWR \leftarrow PWR \cup \{(a_i, a_j, r)\}$;
  end
end

A corner case of this method is represented by attribute pairs that consistently assume the same values. For example, let us consider A, B with $\#_A = (a, a)$ and $\#_B = (a, a)$. Algorithm 1 would return as result $PWR = \{(A, B, 1)\}$, but A, B represent the same information. In order to overcome this problematic case, we restrict the search to those attributes which do not have extreme repetitiveness. Thus, we penalize the both high and low values by comparing the co-repetitiveness by computing the mean value between the individual repetitions of the attributes. Therefore, we select the pairs $r, \overline{r_{A,B}}$, where r was calculated by Algorithm 1 and $\overline{r_{A,B}} = \frac{Rep(A)+Rep(B)}{2}$. Ideally, the best candidate has a low value of r and a high value of $\overline{r_{A,B}}$. If we consider the couples in a Cartesian space, the optimal point has coordinates $Opt = (0, 1)$. This entails, that the best candidates for being case-activity identifier pairs are the geometrically closest to the optimum, i.e., they minimize the Euclidean distance $\delta^{Opt} = \|Opt - (r, \overline{r_{A,B}})\|$. The choice of one or another case identifier with similar low distance δ^{Opt} defines different perspectives on the business process.

Discovering the Process. In Sect. 3.3 we identified pairs that are candidate for being cases and activities. This final step is concerned with extracting a process model out of the event pool PL. This is performed by labeling which attributes of the event pool are cases and which are activities, and order them by timestamp. There are generally multiple candidate pairs returned by the aforementioned step. Thus, the challenge is to select the correct candidate. However, the set of possible candidates can be restricted to a small number of elements, through the usage of a customizable parameter κ, resulting in a set of pairs can be manually inspected. The κ attributes that were identified as *relevant* can contain other information that can be exploited by process mining algorithms, such as [12, 13]. At this point we have identified the cases, the activities, the timestamps and additional information. Thus, we can generate a log file and apply process discovery.

3.4 Example

Here, we see an example of how the method works in practice. For our purpose let us consider the subset of attributes $A = \{trId, mTrId, timestamp, uName, oName\}$ from Table 1.

First, we compute the case and activity candidates. For all the attributes $trId$, $mTrId$, $timestamp$, $uName$, $oName$ we compute $Rep(a)$, $a \in A$. The results are $Rep(trId) = 0.67$, $Rep(mTrId) = 0.67$, $Rep(timestamp) = 0$, $Rep(uName) = 0.67$, $Rep(oName) = 0.78$. Note, that attribute $timestamp$ as it was detected as $timestamp$ because it has no degree of repetitiveness. Thus, we can already exclude $timestamp$ from the case identifier candidates.

Table 2. Pairwise repetitiveness PWR (i.e. co-repetitiveness) and mean co-repetitiveness $\overline{r}$ computed for the example case

	trId	**mTrId**	**timestamp**	**uName**	**oName**
trId PWR, $\overline{r}$	- , -	(0.00, 0.66)*	(0.00, 0.33)	(0.33, 0.66)	(0.44, 0.72)
mTrId PWR, $\overline{r}$	(0.00, 0.67)*	- , -	(0.00, 0.33)	(0.44, 0.66)	(0.44, 0.72)
timestamp PWR, $\overline{r}$	(0.00, 0.33)	(0.00, 0,33)	- , -	(0.00, 0.33)	(0.00, 0.38)
uName PWR, $\overline{r}$	(0.33, 0.66)	(0.44, 0.72)	(0.00, 0.33)	- , -	(0.66, 0.72)
oName PWR, $\overline{r}$	(0.44, 0.72)	(0.44, 0.72)	(0.00, 0.38)	(0.66, 0.72)	- , -

The next step is to compute the co-repetitiveness set, as from Algorithm 1 and obtain the set PWR. We report the values $r \in PWR$ corresponding to each pair in the row PWR of Table 2. Likewise, we also compute the average co-repetitiveness $\overline{r_{A_i,A_j}}$ for all the pairs of attributes and report the result as $\overline{r}$ next to PWR in the same row of the table.

We have finally obtained a set of points $r, \overline{r}$ of the Cartesian space. For each point we compute the Euclidean distance δ^{Opt} to the optimum point $Opt = (0, 1)$. In our example, the couple $(trId, mTrId)$ scores the lowest distance from Opt, i.e., $\delta^{Opt}(trId, mTrId) = 0.33$ (cf. couple marked with an asterisk * in Table 2). This means that the attributes $trId$, $mTrId$ are candidates for being one the case identifier and the other the activity identifier.

4 Evaluation: Industry Use Case

In this section we evaluate our approach against real world data. In particular, we aim at showing the applicability and usefulness of our approach to support selection of case and activity identifiers.

4.1 Experimental Setup and Dataset

We used our approach on the ordering process scenario described in Sect. 2.1. The reference process model is the one shown in figure to Fig. 2, and it describes in a coarse-grained fashion how the process is supposed to be executed. Mechanisms to query events from the middleware were already in place. In particular, we retrieved the data exploiting the search engine Elaticsearch[2]. The output was presented in the JSON[3] format. Afterwards, the different events were grouped together and transformed as in Table 1.

Next, we describe the dataset used for in our evaluation. To extract the process data, we used knowledge about the start and end events. Then we proceeded by querying for all the start events in a specific day. For each start event, we followed several related events and made a new query in the middleware for each of them. This procedure led to systematically obtaining all related events, including several end events.

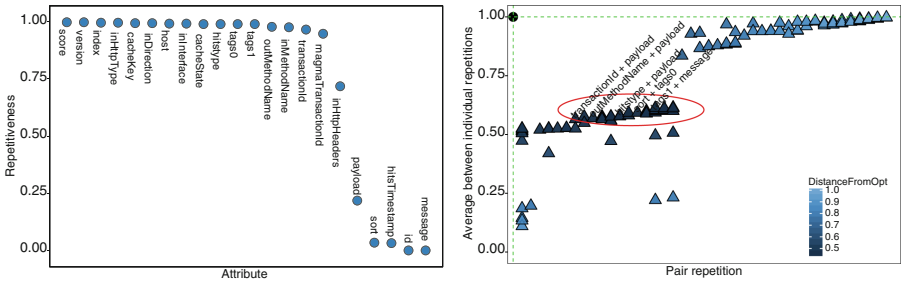
We extracted 8042 low-level event data concerning the ordering process. Note that this is already a considerable amount of data as we focus on the event attributes. In fact, similar results were obtained even with a smaller portion of data. After collecting all the events and building the event pool, we obtained a total number of 41 attributes. The data collection was stored in CSV files which were then also manually checked for parsing errors. Note that, despite their reference to a known start event, the events may and usually cover more than the ordering process. Also, due to the fact that some processes may take weeks before reaching the final state, we are aware that the traces may be incomplete. The aim is to show the applicability of our method and that we can identify meaningful candidates.

4.2 Results

In this section we show the results of applying our approach to the dataset. We proceed following the approach step by step. To this end, we implemented a proof-of-concept prototype[4] using Python and R.

Case and Activity Candidates. Fig. 4a shows the results of the repetitiveness computed for each attribute. Already at this stage events that are not case or activity candidates can be filtered out. In fact, time the attributes *timestamp* and *id* have a repetitiveness close to 0. Figure 4b plots the relation between pairwise mean repetitiveness and the co-repetitiveness of attribute pairs. The x-axis represents the repetitiveness of each pair, whereas the y-axis represents the mean value between the individual repetitiveness degree of each attribute of the pair. The optimal point $Opt = (1, 0)$ is colored in black, the points with a lower distance δ^{Opt} are colored with darker tonality, and the red ellipsis represents the top candidates that score the lowest δ^{Opt}.

[2] https://www.elastic.co/products/elasticsearch.
[3] https://www.json.org.
[4] open source in https://github.com/s41m1r/case-and-activity-identification.



(a) Individual repetitiveness of the event attributes

(b) Results of case and activity identification on the real world data

Fig. 4. Results of the approach

We picked the top $\kappa = 10$ candidate for case and activity. These were the following couples: {(*trId*, *payload*), (*payload*, *host*), (*outMethodName*, *payload*), (*outMethodName*, *mTrId*), (*payload*, *cacheState*), (*cacheKey*, *payload*), (*cacheState*, *payload*), (*payload*, *inDirection*), (*payload*, *inInterface*), (*inDirection*, *payload*), (*inInterface*, *payload*)}. Because our approach is agnostic about the meaning of the attributes, we relied on domain knowledge to select attributes that are more meaningful. As a result, the couple (*outMethodName*, *mTrId*) was chosen as the best candidate, with *mTrId* being chosen as the case and *outMethodName* as the activity.

The company could confirm that *mTrId* and *outMethodName* were indeed key attributes representing respectively a transaction case (i.e., a sequence of actions taken by the customer) and a method called by a service task that implemented the activity. Therefore, the resulting business represents a user transaction perspective on the business process, with each activity being represented the action of the user.

Discovered Process Model. In this step we build an event log including the couple (*outMethodName*, *mTrId*). The timetamp that was identified in the previous phase in Sect. 3.3. In addition, we also included the top most relevant attributes that were connected to *outMethodName* and *mTrId*. In particular, we found other attributes like *inMethodName* and *sort* which were candidates for alternative activity and timestamp, respectively. More specifically, we analyzed the attribute ranges for the most relevant attributes that were identified previously. We report them in Table 3.

Figure 5 shows the resulting business process that was mined using the Celonis[5] process mining tool. The outcome showed that the process is similar to the one presented in Fig. 2, with the real process having a higher number of activities. In particular, Celonis showed that there were 13 cases in our event log and that happy path is constituted by the sequence of activities *ReqProms*, *QueryTarifs*, *ValidateAddress*, *RegCustomer*.

[5] https://www.celonis.com/.

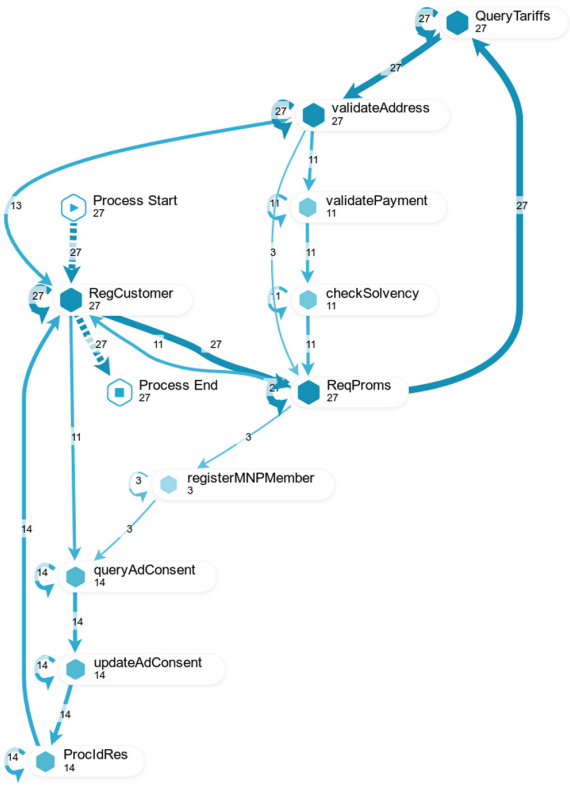


Fig. 5. End-to-end business process mined with Celonis

Table 3. Attribute ranges

Attribute	**id**	**timestamp**	**trId**	**outMethodName**	**tags0**
Range	20 digits	1511996925852	FE_1de0ea5d	qAdConsent	response
	unique	1511996925917	FE_213dc19d	checkSolv	backend
	values	1511996926244	FE_239bc992	RegCustomer	client
		1511996926335	FE_3039d556	/api/ordering	request
		[cont.]	[cont.]	[cont.]	[cont.]
Levels	**835**	**805**	**28**	**12**	**5**
Attribute	**host**	**payload**	**mTrId**	**inMethodName**	**[cont.]**
Range	ip address	29 JSON objects	MA_1ZBLJVV2	validatePayment	
		189 XML objects	MA_4YcwwtvU	checkSolv	
		346 SOAP messages	MA_75FDzegp	RegCustomer	
			MA_9USSsx8Y	/api2/ordering	
			[cont.]	[cont.]	
Levels	**4**	**564**	**31**	**12**	

Discussion. We identified the top candidate for case id and activity. In the top ranking there were also the *outMethodName*, *inMethodName* and *mTrId*. These one were also chosen by the company to implement their own process monitoring tool. The development process monitoring tool also exploits domain knowledge in order to understand whether an events signifies the start or an end of an activity. As a result they were able to map 8 activities over 9 for process monitoring.

Limitations of the approach are related to the quality of data. Given their diverse provenance, the preprocessing step is crucial for eliminating noise and further causes of parsing mistakes. The case and activity identification step of the approach is useful for identifying case-activity pairs. However, it supports no semantic and relies on expert domain knowledge. Lastly, the quality of the model resulting from the last step depends on the process mining algorithm.

5 Conclusion

In this work, we tackle the problem of monitoring business processes from event data which lack a *case* notion. The scenario is present in industry where events from different systems are pooled together. Process monitoring in these scenarios the discovery of the process. We proposed an approach to preprocess the data to solve the heterogeneity problem, and detect cases-activity candidate pairs. We use this information to build an event log and apply a process mining algorithm to obtain a process model. Our approach is suitable for real case scenarios where the event provenance is diverse and no event schema is know a priori. The approach is customizable by the domain engineer and can provide the top κ case-activity identifier candidates. In future work, we aim at improving the approach towards dealing with automatically detecting granularity between events and business activities, the discovery of causal dependencies, and the visualization of links between event pairs.

References

1. de A. R. Gonçalves, J.C., Santoro, F.M., Baião, F.A.: Business process mining from group stories. In: CSCWD. pp. 161–166. IEEE (2009)
2. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, Heidelberg (2016)
3. Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. Softw. Syst, Model **17**(2), 573–598 (2017)
4. Baier, T., Mendling, J., Weske, M.: Bridging abstraction layers in process mining. Inf. Syst. **46**, 123–139 (2014)
5. Baier, T., Rogge-Solti, A., Mendling, J., Weske, M.: Matching of events and activities: an approach based on behavioral constraint satisfaction. In: SAC, pp. 1225–1230. ACM (2015)
6. Bayomie, D., Awad, A., Ezat, E.: Correlating unlabeled events from cyclic business processes execution. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 274–289. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_17
7. Calvanese, Diego, Kalayci, Tahir Emre, Montali, Marco, Tinella, Stefano: Ontology-based data access for extracting event logs from legacy data: the onprom tool and methodology. In: Abramowicz, Witold (ed.) BIS 2017. LNBIP, vol. 288, pp. 220–236. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_16
8. Calvanese, D., Montali, M., Syamsiyah, A., van der Aalst, W.M.P.: Ontology-driven extraction of event logs from relational databases. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 140–153. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_12
9. Conforti, R., Dumas, M., García-Bañuelos, L., Rosa, M.L.: BPMN miner: automated discovery of BPMN process models with hierarchical structure. Inf. Syst. **56**, 284–303 (2016)
10. Di Ciccio, C., Mecella, M.: Mining artful processes from knowledge workers' emails. IEEE Internet Comput. **17**(5), 10–20 (2013)
11. Friedrich, Fabian, Mendling, Jan, Puhlmann, Frank: Process model generation from natural language text. In: Mouratidis, Haralambos, Rolland, Colette (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 482–496. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21640-4_36
12. Günther, Christian W., van der Aalst, Wil M.P.: Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In: Alonso, Gustavo, Dadam, Peter, Rosemann, Michael (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75183-0_24
13. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
14. 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: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 125–141. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_8
15. Motahari-Nezhad, H.R., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. VLDB J. **20**(3), 417–444 (2011)

16. do Nascimento, G.S., Iochpe, C., Thom, L., Kalsing, A.C., Moreira, Á.: Identifying business rules to legacy systems reengineering based on BPM and SOA. In: Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2012. LNCS, vol. 7336, pp. 67–82. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31128-4_6
17. Nooijen, E.H.J., van Dongen, B.F., Fahland, D.: Automatic discovery of data-centric and artifact-centric processes. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 316–327. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_36
18. Raichelson, L., Soffer, P., Verbeek, E.: Merging event logs: combining granularity levels for process flow analysis. Inf. Syst. **71**, 211–227 (2017)
19. Reguieg, H., Benatallah, B., Nezhad, H.R.M., Toumani, F.: Event correlation analytics: scaling process mining using mapreduce-aware event correlation discovery techniques. IEEE Trans. Serv. Comput. **8**(6), 847–860 (2015)
20. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: Intra and inter-case features in predictive process monitoring: a tale of two dimensions. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 306–323. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_18
21. Weber, I., Farshchi, M., Mendling, J., Schneider, J.: Mining processes with multi-instantiation. In: SAC. pp. 1231–1237. ACM (2015)