Federated Process Mining: Exploiting Event Data Across Organizational Boundaries

Wil M.P. van der Aalst
Process and Data Science (Informatik 9), RWTH Aachen University, Aachen, Germany and
Fraunhofer-Institut für Angewandte Informationstechnik (FIT), Sankt Augustin, Germany
Email: wvdaalst@pads.rwth-aachen.de

***Abstract*—Many organizations use process mining to improve their internal processes. However, many processes span multiple organizations, e.g., organizations that form a supply chain or a global production network. In order to improve processes across different parties, one cannot assume a single overall event log. Organizations may not be willing to share event data and may use different identifiers and logging conventions. Federated process mining aims to tackle these problems by creating views on the cross-organizational processes such that analysis is possible. A federated event log transparently maps multiple organization-specific event logs into a single unified representation. Using such a federated event log, it is possible to apply a range of existing process mining techniques and tools. If organizations are not willing to share raw event data, they need to resort to event log abstractions. In such settings, federated process mining approaches merge these abstractions to create an overall view on the process. This paper provides a framework to reason about the different forms of federated process mining enabling inter-organizational process transparency and improvement.**

***Index Terms*—process mining, event data, data management, object-centric processes, supply chains, production networks**

I. Introduction

Process mining research started in the late 1990-ties [1] as a response to problems related to process modeling and process automation. Many organizations that tried to adopt Workflow Management (WFM) or Business Process Management (BPM) systems experienced major problems. WFM/BPM technology was expensive and doomed to fail when organizations used idealized models to configure them [21]. Process mining uses event data to learn and improve the real processes [1]. Discovered process models tend to be very different from hand-made models. Conformance checking techniques can be used to detect and explain deviations between models and reality. Process mining is *evidence-based* and complements preexisting process management practices.

Currently, there are over 35 providers of commercial process mining tools (e.g., Celonis, Fluxicon, Minit, Mehrwerk, MyInvenio, PAFnow, Signavio, QPR, and many more) and several open-source initiatives (e.g., ProM, PM4Py, RapidProM, and BupaR). The adoption of process mining in practice increased rapidly in recent years [15]. The main reasons to use process mining are process transparency, process improvement, cost reduction, compliance, throughput time reduction, and digital transformation [10]. However, most applications of process mining are *intra-organizational*. Typical use cases are found in procurement, sales, accounting, logistics, production, customer service, payments, controlling, after-sales, and marketing [10]. Process mining is rarely applied in an *inter-organizational* setting. This is mostly caused by confidentiality concerns and interoperability problems. Given the huge potential of process mining as a tool for improving cross-organizational processes, we present an initial framework for *federated process mining*.

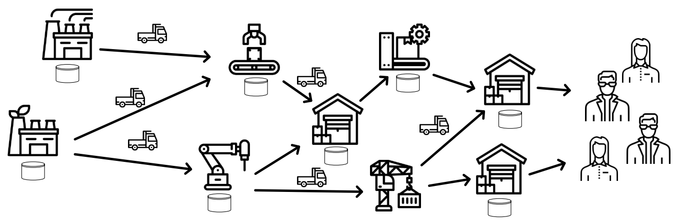
Fig. 1. A network of organizations collecting event data while engaging in cross-organizational processes.

Figure 1 sketches the problem we are trying to address. Consider, for example, a complex supply chain or a global production network involving multiple organizations. To get an idea of the complexity of such networks, consider the supply pyramid of a car manufacturer like BMW. BMW is a so-called OEM (Original Equipment Manufacturer) which gets system parts from so-called Tier-1 suppliers such as Bosch, Continental, Denso, Magna, and ZF. These Tier-1 suppliers get components from Tier-2 suppliers, which in turn may get smaller parts from Tier-3 suppliers. A single BMW may have over 30,000 parts, counting every part down to the smallest screws. BMW's global production network consists of 3,200 suppliers at 4,500 production locations in more than 50 countries. In Figure 1, all organizations store their own event data. The challenge is to monitor and improve the *overall* process.

The term *federated process mining* refers to solutions that transparently map event data from multiple autonomous data sources into a single federated data source *with the goal to monitor, analyze, and improve cross-organizational processes*. The idea is similar to creating a federated database system composed of a collection of cooperating database systems that are autonomous and possibly heterogeneous [22]. The goal of a federated database system is to provide a unified view, hiding irrelevant technical details. In federated process mining, we have the additional challenge that event data are potentially highly sensitive. A Tier-1 supplier may not want to reveal its internal processes to the OEM. However, there are also many similarities. For example, different organizations may use different identifiers that refer to the same object or concept.

We consider *two types of federated process mining*. Both start from the assumption that we have n organizations that collect event data about one or multiple processes. $C = \{L_1, L_2, \ldots, L_n\}$ is an event log collection where L_i is the event log of organization $i \in \{1, \ldots, n\}$. Event data may be stored using standard formats like XES [14] or OCEL [13]. However, existing approaches assume a single event log and not a collection of events logs. How to analyze the overall process(es) starting from C?

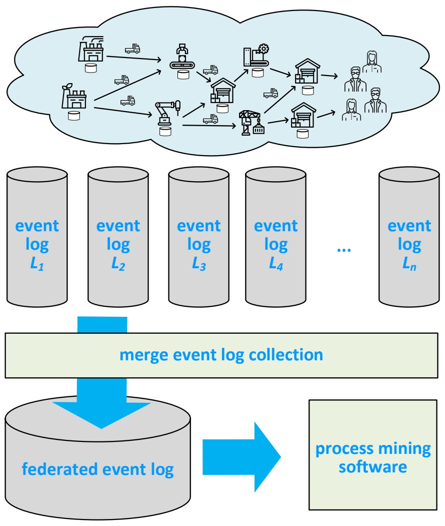

Fig. 2. A network of organizations collecting event data while engaging in cross-organizational processes (Type I - federated process mining).

The *first type* of federated process mining is depicted in Figure 2. Here the assumption is that the organizations are willing to share event data. Note that these data may have been filtered to hide certain aspects. The goal is to create a federated event log. This involves several challenges, as will be explained later.

The *second type* of federated process mining is depicted in Figure 3. Here the assumption is that the organizations are *not* willing to share event data. Instead, only *abstractions* can be shared [2]. For example, the different organizations may be willing to share so-called Directly-Follows Graphs (DFGs) [3]. A DFG only shows aggregate information and cannot be used to reconstruct individual process instances.

This paper discusses both types of federated process mining and introduces a formal framework to reason about process mining in a cross-organizational setting. The remainder is organized as follows. Section II introduces basic notations for process mining in general and event data in particular. We rely on so-called *Object-Centric Event Logs* (OCEL) [13] to ensure flexibility and handle divergence and convergence in processes [4]. Section III defines the problem starting from a collection of OCEL-based data sources. Section IV discusses the first

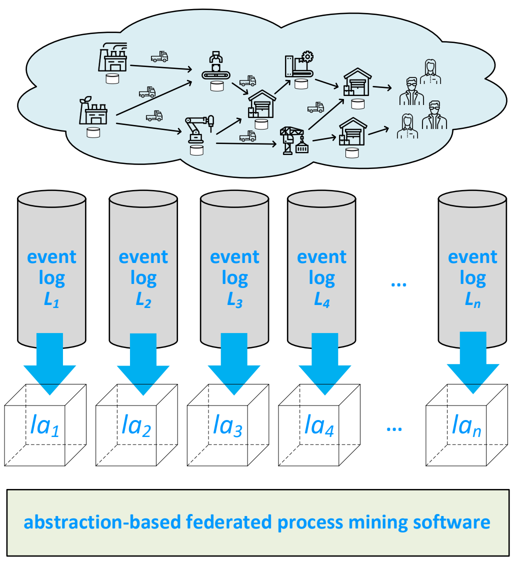

Fig. 3. A network of organizations sharing only abstractions of their event data (Type II - federated process mining).

type of federated process mining which aims at creating a unified federated event log. The second type of federated process mining is discussed in Section V. In this scenario only process-based abstractions [2] are shared. Section VI concludes the paper.

II. Event Data

Event data provide the starting point for *process mining*. Using *process discovery* techniques, event data can be used to uncover the actual processes. Using *conformance checking*, differences and commonalities between modeled and real processes can be analyzed. Process discovery and conformance checking provide the basis for advanced forms of process mining such as *comparative process mining* (understanding performance differences between two processes or the same process in different periods), *predictive process mining* (using the discovered models to predict performance and compliance problems), and *action-oriented process mining* (turning insights into actions).

In this paper, we use so-called *object-centric event logs* based on the *OCEL* (Object-Centric Event Log) standard [13], see ocel-standard.org for details and examples. To keep the notations as simple as possible, we use a simplified formalization that captures the essence of OCEL. The reason for using OCEL rather than XES (http://xes-standard.org/) is that in cross-organizational settings, it is often unclear what the case notion should be. In [4], we elaborate on the need for such a format. We cannot assume that there is a single case notion, instead we use multiple case notions (called object types) and each event may refer to *any number of objects* corresponding to *different object types*.

One can convert an object-centric event log into a standard event log (e.g., in XES format) by picking an object type and replicating each event for each object of the selected type. The resulting event log is called the *flattened* event log. This approach is often used and works well, however, also has three possible problems: (1) *deficiency*, (2) *convergence*, and (3) *divergence*. Deficiency means that events in the original event log have no corresponding events in the flattened event log. Hence, events may disappear from the data set. Convergence refers to the fact that events referring to multiple objects of the selected type are replicated, possibly leading to unintentional duplication. For example, when computing costs or resource usage such replicated events can be very misleading. Divergence points to the more subtle problem that events referring to different objects of a type not selected as the case notion are considered to be causally related. For example, two events refer to the same purchase order but different products, or two events refer to the same patient but different blood samples. This creates loops that do not really exist (interleaving of causally unrelated events). To avoid such problems, we developed techniques for *object-centric process mining*, e.g., learning object-centric Petri nets that do not have the three problems mentioned [5]. Although this is outside the scope of this paper, it is important to note that, in a cross-organizational setting, we cannot assume a single case notion or start from a flattened event log.

First, we define universes for event identifiers, object identifiers, attribute names, attribute values, activities, timestamps, attribute name value mappings, and object types. Activities and timestamps can be seen as special values.

Definition 1 (Universes): $\mathcal{E}$ is the universe of event identifiers, $\mathcal{O}$ is the universe of object identifiers (such that $\mathcal{E} \cap \mathcal{O} = \emptyset$), $\mathcal{N}$ is the universe of attribute names, $\mathcal{V}$ is the universe of attribute values, $\mathcal{A} \subseteq \mathcal{V}$ is the universe of activities, $\mathcal{T} \subseteq \mathcal{V}$ is the universe of timestamps (totally ordered), $\mathcal{M} = \mathcal{N} \not\rightarrow \mathcal{V}$ is the set of attribute name value mappings, and $\mathcal{OT}$ is the universe of object types.

An attribute name value mapping $f \in \mathcal{M} = \mathcal{N} \not\rightarrow \mathcal{V}$ is a partial function mapping attribute names onto attribute values. This allows us to compactly define a *simplified object-centric event log*. Definition 2 captures the essence of the OCEL (Object-Centric Event Log) standard [13], see ocel-standard.org. A simplified object-centric event log refers to a collection of related events and objects. Each event has an activity name and timestamp. Next to the timestamps there may be a partial order that may be stricter or more liberal, but should not be conflicting.

Definition 2 (Simplified Object-Centric Event Logs): A simplified object-centric event log $L = (E, \prec, O, R, \pi)$ is composed of a set of events $E \subseteq \mathcal{E}$, a strict partial order $\prec \subseteq E \times E$,[1] a set of objects $O \subseteq \mathcal{O}$, a relation $R \subseteq E \times O$, a function $\pi \in (E \cup O) \rightarrow \mathcal{M}$ such that for any $e \in E$: $\{act, time\} \subseteq dom(\pi(e))$, $\pi(e)(act) \in \mathcal{A}$, and $\pi(e)(time) \in \mathcal{T}$, for any $o \in O$: $\{type\} \subseteq dom(\pi(o))$, $\pi(o)(type) \in \mathcal{OT}$. $\mathcal{L}$ is the universe of simplified object-centric event logs.

[1]A strict partial order is irreflexive, transitive, and asymmetric. Hence, for any $e, e_1, e_2, e_3 \in E$: $e \not\prec e$ (irreflexivity), if $e_1 \prec e_2$ and $e_2 \prec e_3$, then $e_1 \prec e_3$ (transitivity), and if $e_1 \prec e_2$, then $e_2 \not\prec e_1$ (asymmetry).

To improve readability, we use the notation $\pi_x(e) = \pi(e)(x)$ and $\pi_y(e) = \pi(o)(y)$ to refer to an event attribute x of e and an object attribute y of o. For example, $\pi_{act}(e) = \pi(e)(act)$ is the activity performed by event e, $\pi_{time}(e) = \pi(e)(time)$ is the timestamp of event e, $\pi_{type}(o) = \pi(o)(type)$ is the object type of object o. Instead of writing "simplified object-centric event log", we simply refer to $L \in \mathcal{L}$ as an *event log*, knowing that it is not a conventional event log.

Consider an example event log $L = (E, \prec, O, R, \pi)$ with:

- $E = \{e_1, e_2, e_3, e_4, \ldots\}$ is the set of events,
- $\prec = \{(e_1, e_2), (e_1, e_3), (e_1, e_4), \ldots\}$ is the strict partial order,
- $O = \{o_1, o_2, o_3, o_4, \ldots\}$ is the set of objects,
- $R = \{(e_1, o_1), (e_1, o_2), (e_1, o_3), (e_1, o_4), (e_2, o_2), (e_3, o_3), (e_4, o_4), \ldots\}$ relates events and objects,
- $dom(\pi(e_1)) = dom(\pi(e_2)) = dom(\pi(e_3)) = dom(\pi(e_4)) = \{act, time, \ldots\}$,
- $dom(\pi(o_1)) = \{type, price, \ldots\}$ and $dom(\pi(o_2)) = dom(\pi(o_3)) = dom(\pi(o_4)) = \{type, prod, \ldots\}$,
- $\pi_{act}(e_1) = place\_order$, $\pi_{act}(e_2) = \pi_{act}(e_3) = \pi_{act}(e_4) = pick\_item$, etc.
- $\pi_{time}(e_1) = 14{:}23$, $\pi_{time}(e_2) = 14{:}55$, $\pi_{time}(e_3) = 15{:}05$, $\pi_{time}(e_4) = 15{:}10$, etc.
- $\pi_{type}(o_1) = order$, $\pi_{type}(o_2) = \pi_{type}(o_3) = \pi_{type}(o_4) = item$, etc.
- $\pi_{price}(o_1) = \$825.23$, $\pi_{prod}(o_2) =$ iPhone, $\pi_{prod}(o_3) =$ iPad, $\pi_{prod}(o_4) =$ iPod, etc.

Note that event e_1 refers to one order and three items (one iPhone, one iPad, and one iPod). Events e_2, e_3, and e_3 refer to a single item.

To be as general as possible, events have timestamps and are partially ordered. Note that $\prec$ may express causalities. Also, timestamps may be coarse-granular, e.g., just a date (or even a week number or year). Therefore, two events that happen on the same day may have the same timestamp. It is also possible that events are not ordered according to $\prec$. In fact, it is possible that $\prec = \emptyset$ (i.e., no ordering relations). The other extreme is that $\prec$ is a strict total ordering. We typically require that both ordering relations (time-based and $\prec$) are not conflicting, leading to the notion of consistency.

Definition 3 (Consistent): Let $L = (E, \prec, O, R, \pi) \in \mathcal{L}$ be an event log. L is consistent if for any $e_1, e_2 \in E$ such that $e_1 \prec e_2$: $\pi_{time}(e_1) \leq \pi_{time}(e_2)$ (i.e., partial order and timestamps do not disagree). L is time-constrained if for any $e_1, e_2 \in E$: $e_1 \prec e_2 \Rightarrow \pi_{time}(e_1) < \pi_{time}(e_2)$. L is order-constrained if for any $e_1, e_2 \in E$: $\pi_{time}(e_1) < \pi_{time}(e_2) \Rightarrow e_1 \prec e_2$.

As mentioned earlier, event logs can be flattened by picking an object type as a case notion.

Definition 4 (Flattened Event Log): Let $L = (E, \prec, O, R, \pi) \in \mathcal{L}$ be an event log and $ot \in \mathcal{OT}$ be an object type. $O^{ot} = \{o \in O \mid \pi_{type}(o) = ot\}$ are the objects of type ot. $L^{ot} \in O^{ot} \rightarrow E^*$ maps each object of type ot onto a sequence of events such that for any $o \in O^{ot}$ and $L^{ot}(o) = \sigma^o = \langle e_1, e_2, \ldots, e_{|\sigma^o|} \rangle$ the following holds $\{e_1, e_2, \ldots, e_{|\sigma^o|}\} = \{e \in E \mid (e, o) \in R\}$, and for any $1 \leq$

$i < j \leq |\sigma^o|$: $e_i \neq e_j$, $e_i \not\succ e_j$, and $\pi_{time}(e_i) \leq \pi_{time}(e_j)$. L^{ot} is an ot-flattened event log of L.

A flattened event log L^{ot} maps each object o onto a sequence of events $L^{ot}(o)$. This implies that within such a sequence, events are totally ordered. Since the timestamps and ordering relation $\prec$ only provide a partial order, this total order may be non-deterministic.

Using our example event log and picking object type $item$. $O^{item} = \{o_2, o_3, o_4, \ldots\}$, $L^{item}(o_2) = \langle e_1, e_2, \ldots \rangle$, $L^{item}(o_3) = \langle e_1, e_3, \ldots \rangle$, $L^{item}(o_4) = \langle e_1, e_4, \ldots \rangle$, etc.

To apply standard process mining techniques (e.g., basic control-flow discovery), we aim for an event log that is represented as a *multiset of traces* (i.e., sequences of activities). Therefore, we map events onto their activity names.

Definition 5 (Log Simplification): Let $L = (E, \prec, O, R, \pi) \in \mathcal{L}$ be an event log and $L^{ot} \in O^{ot} \rightarrow E^*$ an ot-flattened event log of L. $\Pi^{ot}_{act}(L) = [\pi_{act}(L^{ot}(o)) \mid o \in O^{ot}]$ is a multiset of activity sequences with $\pi_{act}(\langle e_1, e_2, \ldots, e_n \rangle) = \langle \pi_{act}(e_1), \pi_{act}(e_2), \ldots, \pi_{act}(e_n) \rangle$, i.e., each event is replaced by the corresponding activity.

Using our example event log, we get $\Pi^{item}_{act}(L) = [\langle place\_order, pick\_item, \ldots \rangle, \langle place\_order, pick\_item, \ldots \rangle, \langle place\_order, pick\_item, \ldots \rangle, \ldots]$. Note that in such a multiset the same trace may appear many times.

There exist a few process mining techniques that work directly on object-centric event logs [5]. However, the majority of tools and techniques require flattened event logs. Hence, one should be aware of possible interpretations problems due to divergence and convergence [5].

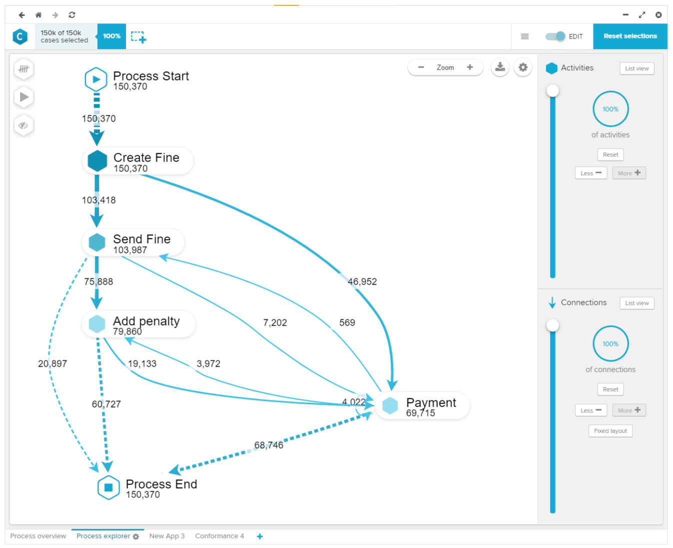

Fig. 4. DFG discovered by Celonis for the Road Traffic Fine Management (RTFM) event log in [8].

To illustrate what one can do with flattened event logs, we use a publicly available event log [8] taken from an Italian information system managing road traffic fines. The event log has 561,470 events containing information about 150,370 traffic violations. Figure 4 shows a Directly-Follows Graph (DFG) generated by Celonis (www.celonis.com). The same graph could have been generated by any of the commercial and open-source systems mentioned earlier (e.g., ProM, Disco, Minit, MyInvenio, PAFnow, and QPR).

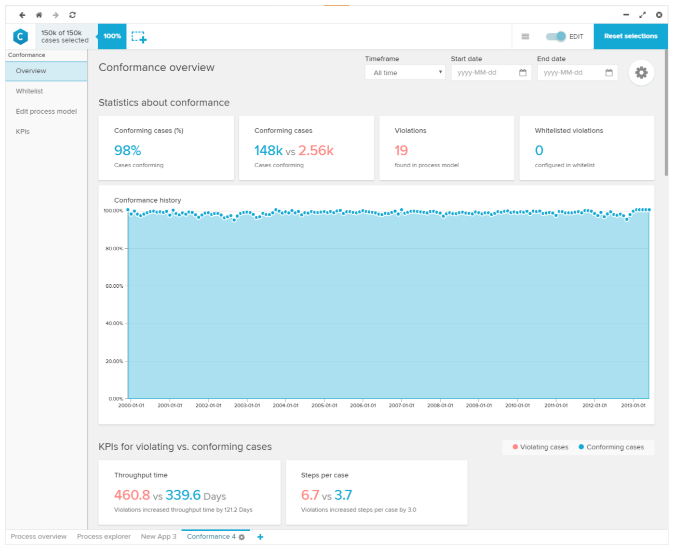

Fig. 5. Conformance checking results for the RTFM event log in [8] and the BPMN model in Figure 6.

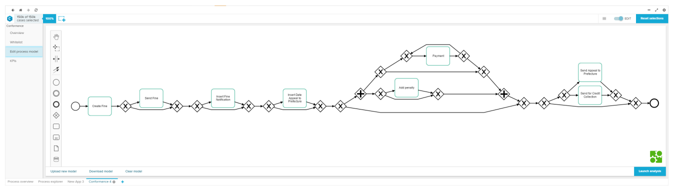

Fig. 6. BPMN model discovered using the inductive miner in Celonis for the RTFM event log in [8].

Figure 5 shows conformance diagnostics for the event log and the BPMN model shown in Figure 6. Celonis discovered 19 unique deviations between the model and log. 98% of the cases are conforming. The 2560 non-conforming cases take, on average, 121 days longer and have three more steps. Detailed diagnostics are provided for each deviation. The process model in Figure 6 was discovered using the inductive miner [16] in Celonis based on the most frequent variants. To check conformance, we could also have modeled the process by hand.

Figures 4, 5, and 6 illustrate that process mining can be used to discover processes, check compliance, analyze bottlenecks, etc. Process mining results can be particularly valuable in processes crossing organizational boundaries. However, this is problematic, as mentioned in the introduction. Therefore, we elaborate on techniques for federated process mining.

III. Event Log Collection

Figures 2 and 3 illustrate the problem we are trying to address in this paper. There are n organizations each logging events. How to use a heterogeneous collection of n local event logs to make statements about the overall process?

Definition 6 (Event Log Collection): An event log collection $C = \{L_1, L_2, \dots, L_n\} \subseteq \mathcal{L}$ is a set of event logs such that the sets of events are pairwise disjoint. $L_i = (E_i, \prec_i, O_i, R_i, \pi_i)$ refers to the i-th object-centric event log and for any $1 \leq i < j \leq n$: $E_i \cap E_j = \emptyset$.

We assume that all local events have a unique identifier, i.e., an event identifier can only appear in one event log. Object identifiers may be shared among different event logs. Given an event log collection $C = \{L_1, L_2, \dots, L_n\}$, we would like to make statements about cross-organizational processes. For example, what is the cause of a new bottleneck or what will the resource load be tomorrow. Without information sharing, organization i can only see L_i whereas performance and compliance problems may be caused by behavior in L_j (with $j \neq i$).

In this paper, we consider *two types* of federated process mining. The first type (depicted in Figure 2) creates a federated event log by merging the different event logs. The second type of federated process mining (depicted in Figure 3) uses abstractions, i.e., the different parties do not share events, but use aggregated statements about processes like DFGs or similar. These abstractions are merged to make statements about the overall processes.

Before we discuss possible solutions, we first mention some of the main challenges.

- The different event logs may use *different object identifiers*. Object linkage (also known as record linkage, data matching, entity resolution, etc.) is the task of finding objects in different data sources that refer to the same object.
- We assume that each event appears in only one event log (unique identifiers). However, recorded events on the interface of two or more organizations may refer to the *same logical event* (e.g., a send event and a receive event that refer to the same message of a procedure call recorded by the caller and callee).
- The different organizations may use *different clocks* and record at *different levels of granularity*. For example, one organziation only records dates whereas another organization records events with millisecond precision.
- There may be many *confidentiality* considerations preventing organizations from sharing data. As a result, events may be suppressed or other confidentiality preserving operations may be performed (e.g., adding noise) [19], [20].

Some initial work has been done on applying process mining in an inter-organizational setting. In [9], we presented the EDImine Framework for enabling the application of process mining techniques in the field of EDI-supported inter-organizational business processes, and for supporting inter-organizational performance evaluation using business information from EDI messages, event logs, and process models. In [17], we describe seven basic patterns to capture modeling concepts that arise commonly in supply chains. These patterns can be combined to build an overall Petri net that can be analyzed using dependency graphs and simulation. The paper [18], was the first work considering the application of process mining in supply chains. In [11] a standard for processing Radio Frequency Identification (RFID) events was proposed to make supply chain data accessible for process mining. The same authors provide a more detailed paper [12] focusing on different case notions in supply chains where objects are grouped (e.g., items into packages and packages into containers). However, none of the approaches mentioned addresses the problem of federated process mining in a comprehensive manner. Also, there has been tremendous progress in process mining, not only in terms of techniques for process discovery and conformance checking, but also in new logging formats like OCEL. Therefore, this paper provides a framework for federated process mining starting from OCEL-like event logs.

In the remainder, we discuss the two types of federated process mining. Section IV shows how to create a unified event log and Section V assumes that parties only share process-based abstractions rather than event logs.

IV. Creating a Federated Event Log

As shown in Figure 2, the first type of federated process mining creates a *federated event log*. The naive approach would be to simply merge the n event logs. However, the organizations may use different identifiers to refer to the same object or events. Hence, we use a mapping γ_C to relate both.

Definition 7 (Federated Event Log): Let $C = \{L_1, L_2, \dots, L_n\} \subseteq \mathcal{L}$ be an event log collection with $L_i = (E_i, \prec_i, O_i, R_i, \pi_i)$ for $1 \leq i \leq n$. $E = \bigcup_{1 \leq i \leq n} E_i$ and $O = \bigcup_{1 \leq i \leq n} O_i$. Let $\gamma_C \in (E \cup O) \not\rightarrow ((E \cup O) \setminus \mathit{dom}(\gamma_C))$. γ_C is a partial function, i.e., $\mathit{dom}(\gamma_C) \subseteq E \cup O$. We require that for any $e \in \mathit{dom}(\gamma_C) \cap E$: $\gamma_C(e) \in E$, and for any $o \in \mathit{dom}(\gamma_C) \cap O$: $\gamma_C(o) \in O$.

Given γ_C, we create the federated event log $L_{\gamma_C} = (E', \prec', O', R', \pi') \in \mathcal{L}$ with

- $E' = E \setminus \mathit{dom}(\gamma_C)$,
- $\tilde{\gamma}_C(x) = \gamma_C(x)$ if $x \in \mathit{dom}(\gamma_C)$ and $\tilde{\gamma}_C(x) = x$ if $x \notin \mathit{dom}(\gamma_C)$,
- $\prec' = \{(\tilde{\gamma}_C(e_1), \tilde{\gamma}_C(e_2)) \mid \exists_{1 \leq i \leq n} \exists_{(e_1, e_2) \in E}\ e_1 \prec_i e_2\}^+$,
- $O' = O \setminus \mathit{dom}(\gamma_C)$,
- $R' = \{(\tilde{\gamma}_C(e), \tilde{\gamma}_C(o)) \mid (e, o) \in R\}$, and
- $\pi' \in (E' \cup O') \to \mathcal{M}$ such that for any $x \in E' \cup O'$: $\pi'(x) = \bigoplus_{1 \leq i \leq n} \bigoplus_{y \in E_i \cup O_i \mid \gamma_C(y) = x} \pi_i(y)$.[2]

Mapping γ_C plays a key role in Definition 7. Events in $E'' = E \cap \mathit{dom}(\gamma_C)$ and objects in $O'' = O \cap \mathit{dom}(\gamma_C)$ do not appear in the federated event log. Only the events $E' = E \setminus \mathit{dom}(\gamma_C)$ and objects $O' = O \setminus \mathit{dom}(\gamma_C)$ remain. However, properties of the events in E'' and objects O'' are transferred to their "surviving counterparts". This includes the attributes (π') and the event-object relationships (R').

$\prec'$ merges the relations in $\prec_i$ and takes the transitive closure of the strict partial orders. Due to this and the mapping γ_C, the result may not be a strict partial order (e.g., a cycle is

[2]Let f_1 and f_2 be two functions, $f = f_1 \oplus f_2$ merges both such that $\mathit{dom}(f) = \mathit{dom}(f_1) \cup \mathit{dom}(f_2)$ and for any $x \in \mathit{dom}(f_1) \setminus \mathit{dom}(f_2)$: $f(x) = f_1(x)$, and $x \in \mathit{dom}(f_2)$: $f(x) = f_2(x)$. $\bigoplus_{1 \leq i \leq n} f_i = f_1 \oplus f_2 \oplus \dots \oplus f_n$.

introduced). Therefore, there may be different ways of merging $\prec_i$ with $1 \leq i \leq n$ into $\prec'$ such that the results is again a strict partial order.

Similar challenges need to be addressed with respect to time. The individual event logs may be consistent, but consider time at different levels of granularity. One organization may record time in days, the other in hours, and the third one in seconds. When merging the event data, this may lead to inconsistencies. This can be addressed by choosing the coarsest level of time granularity and further strengthen the strict partial orders to not lose information.

Definition 7 does not allow for the creation of new events or objects. Of course, one can also think of the addition of new events or objects. Moreover, events or objects can also be discarded completely (without mapping them). However, the main idea of the first type of federated process mining should be clear: An event log collection $C = \{L_1, L_2, \ldots, L_n\}$ is turned into an overall object-centric event log L.

Such an overall object-centric event log L may be *materialized* (i.e., a new event log is created and stored) or only exist as a *virtual event log* like in a federated database system [22].

V. Federated Process Mining Using Abstractions

Often, it is unfeasible to share event data, e.g., for confidentiality reasons. In such settings, organizations may still share abstractions. This is similar to using *multi-echelon inventory models* in logistics [7]. Multi-echelon inventory models use aggregated inventory levels across the supply chain taking into account stocks at other echelons. These models are used to determine safety stock buffers across the entire supply chain, taking into account aggregated information on downstream and upstream inventory levels. For example, low inventory levels in a central warehouse need to be considered in relation to inventory levels in warehouses closer to the customer. There is no need to increase stock when downstream stock is accumulating. In such multi-echelon inventory models there is no need to know the unique identifiers of products, only quantities suffice.

Such ideas can also be applied to cross-organizational process mining. Instead of sharing event data, one can use abstractions [2].

Definition 8 (Log Abstraction): $\mathcal{LA}$ is the universe of log abstractions (e.g., log statistics, DFGs, footprints, etc.). An abstraction function $abs \in \mathcal{L} \rightarrow \mathcal{LA}$ is a function that maps event logs onto an abstraction. For any $la \in \mathcal{LA}$: $abs^{-1}(la) = \{L \in \mathcal{L} \mid abs(L) = la\}$ is the set of all event logs having the same abstraction la.

In [2] the notion of event log and process model abstraction was discussed in detail. Three characteristic abstractions used in the context of process mining are: Directly-Follows Graphs (DFGs), transition systems (and other types of automata), and cardinality constraint models [2]. For example, Figure 4 shows a DFG generated by Celonis based on 561,470 events related to 150,370 traffic fines [4]. There are many event logs that return the DFG shown in Figure 4. This is the idea of $abs^{-1}(la)$: $L \in abs^{-1}(la)$ if and only if $abs(L) = la$. This shows that the DFG provides information without relating it to individual fines.

To construct a DFG, we need to pick a case notion (i.e., flatten the object-centric event log). In case of multiple object types, the abstraction could be an *array of DFGs* (one for each object type) or higher-level representations as described in [5].

An *abstraction collection* is a set of abstractions in the sense of [2]. Just like in multi-echelon inventory models, the goal is to only share aggregated information and still make informed decisions.

Definition 9 (Abstraction Collection): Let $C = \{L_1, L_2, \ldots, L_n\} \subseteq \mathcal{L}$ be an event log collection with $L_i = (E_i, \prec_i, O_i, R_i, \pi_i)$ for $1 \leq i \leq n$. $A = \{la_1, la_2, \ldots, la_n\}$ is a set of abstractions such that $la_i = abs(L_i)$.

For the second type of federated process mining, depicted in Figure 3, we merge the different abstractions.

Definition 10 (Merging Abstractions): Let $A = \{la_1, la_2, \ldots, la_n\}$ be an abstraction collection based on n event logs. $merge(A) \in \mathcal{LA}$ is an overall abstraction obtained by merging the individual abstractions.

How this is done is a topic of ongoing research. There is clearly a need for dedicated abstraction-based federated process mining software (cf. Figure 3). Since $merge(A) \in \mathcal{LA}$ and $abs^{-1}(merge(A))$ characterizes the set of all possible federated event logs, both types of federated process mining can be related.

VI. Conclusion

This paper introduced the notion of *federated process mining* and provided a formal framework to reason about cross-organizational process mining. We discussed the challenges and identified two different types of federated process mining: (1) approaches where we create a federated event log and (2) approaches based on merging abstractions without sharing event data. The topic is also related to the so-called *Internet-of-Production* (IoP) developed in Aachen as part of the German Excellence Strategy [6]. Within IoP, process mining plays a key role to create so-called “digital shadows” that can be used to analyze and improve operational processes. Thus far, the focus within IoP was on intra-organizational operational processes. However, as discussed in this paper, the importance of *networks* of organizations and processes is rising. Therefore, we would like to develop dedicated techniques and tools for both types of federated process mining.

Acknowledgment

The author thanks the Alexander von Humboldt (AvH) Stiftung and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy–EXC-2023 Internet of Production (390621612) for supporting our research.

References

[1] W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.

[2] W.M.P. van der Aalst. Process Discovery from Event Data: Relating Models and Logs Through Abstractions. *WIREs Data Mining and Knowledge Discovery*, 8(3), 2018.

[3] W.M.P. van der Aalst. A Practitioner's Guide to Process Mining: Limitations of the Directly-Follows Graph. In *International Conference on Enterprise Information Systems (Centeris 2019)*, volume 164 of *Procedia Computer Science*, pages 321–328. Elsevier, 2019.

[4] W.M.P. van der Aalst. Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data. In P.C. Ölveczky and G. Salaün, editors, *Software Engineering and Formal Methods (SEFM 2019)*, volume 11724 of *Lecture Notes in Computer Science*, pages 3–25. Springer-Verlag, Berlin, 2019.

[5] W.M.P. van der Aalst and A. Berti. Discovering Object-Centric Petri Nets. *Fundamenta Informaticae*, 175(1-4):1–40, 2020.

[6] W.M.P. van der Aalst, T. Brockhoff, A. Farhang, M. Pourbafrani, M.S. Uysal, and S. J. van Zelst. Removing Operational Friction Using Process Mining: Challenges Provided by the Internet of Production (IoP). In S. Hammoudi and C. Quix, editors, *Data Management Technologies and Applications (DATA 2020)*, volume 1446 of *Communications in Computer and Information Science*, pages 1–31. Springer-Verlag, Berlin, 2021.

[7] T de Kok, C. Grob, M. Laumanns, S. Minner, J. Rambau, and K. Schade. A Typology and Literature Review on Stochastic Multi-Echelon Inventory Models. *European Journal of Operational Research*, 269(3):955–983, 2018.

[8] M. de Leoni and F. Mannhardt. Road Traffic Fine Management Process (4TU.ResearchData). https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5, 2015.

[9] R. Engel, W. Krathu, M. Zapletal, C. Pichler, J.C. Bose, W.M.P. van der Aalst, H. Werthner, and C. Huemer. Analyzing Inter-Organizational Business Processes. *Information Systems and e-Business Management*, 14(3):577–612, 2016.

[10] G. Galic and M. Wolf. *Global Process Mining Survey 2021: Delivering Value with Process Analytics - Adoption and Success Factors of Process Mining*. Deloitte, 2021. https://www2.deloitte.com/de/de/pages/finance/articles/global-process-mining-survey-2021.html.

[11] K. Gerke, A. Claus, and J. Mendling. Process Mining of RFID-Based Supply Chains. In *IEEE Conference on Commerce and Enterprise Computing*, pages 285–292, 2009.

[12] K. Gerke, J. Mendling, and K. Tarmyshov. Case Construction for Mining Supply Chain Processes. In A. Abramowicz, editor, *Business Information Systems (BIS 2009)*, volume 21 of *Lecture Notes in Business Information Processing*, pages 181–192. Springer-Verlag, Berlin, 2009.

[13] A.F. Ghahfarokhi, G. Park, A. Berti, and W.M.P. van der Aalst. OCEL Standard. www.ocel-standard.org, 2021.

[14] IEEE Task Force on Process Mining. XES Standard Definition. www.xes-standard.org, 2016.

[15] M. Kerremans. Gartner Market Guide for Process Mining, Research Note G00733123. www.gartner.com, 2020.

[16] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, 17(2):599–631, 2018.

[17] E. Liu, A. Kumar, and W.M.P. van der Aalst. A Formal Modeling Approach for Supply Chain Event Management. *Decision Support Systems*, 43(3):761–778, 2007.

[18] L. Maruster, J.C. Wortmann, A.J.M.M. Weijters, and W.M.P. van der Aalst. Discovering Distributed Processes in Supply Chains. In H. Jagdev, J.C. Wortmann, and H.J. Pels, editors, *Collaborative Systems for Production Management*, pages 219–243. Elsevier Science Publishers, Amsterdam, 2003.

[19] M. Rafiei, M. Wagner, and W.M.P. van der Aalst. TLKC-Privacy Model for Process Mining. In F. Dalpiaz, J. Zdravkovic, and P. Loucopoulos, editors, *International Conference on Research Challenges in Information Science (RCIS 2020)*, volume 385 of *Lecture Notes in Business Information Processing*, pages 398–416. Springer-Verlag, Berlin, 2020.

[20] M. Rafiei, L. von Waldthausen, and W.M.P. van der Aalst. Supporting Confidentiality in Process Mining Using Abstraction and Encryption. In P. Ceravolo, M. van Keulen, and M.T. Gomez Lopez, editors, *Postproceedings International Symposium on Data-driven Process Discovery and Analysis*, volume 379 of *Lecture Notes in Business Information Processing*, pages 101–123. Springer-Verlag, Berlin, 2020.

[21] H.A. Reijers, I.T.P. Vanderfeesten, and W.M.P. van der Aalst. The Effectiveness of Workflow Management Systems: A Longitudinal Study. *International Journal of Information Management*, 36(1):126–141, 2016.

[22] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.