Collaboration Miner: Discovering Collaboration Petri Nets

Janik-Vasily Benzin[1] (✉) and Stefanie Rinderle-Ma[1]

Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany
{janik.benzin,stefanie.rinderle-ma}@tum.de

Abstract. Most existing process discovery techniques aim to mine models of process orchestrations that represent behavior of cases within one business process. Collaboration process discovery techniques mine models of collaboration processes that represent behavior of collaborating cases within multiple process orchestrations that interact via collaboration concepts such as organizations, agents, and services. While workflow nets are mostly mined for process orchestrations, a standard model for collaboration processes is missing. Hence, in this work, we rely on the newly proposed collaboration Petri nets and show that in combination with the newly proposed Collaboration Miner (CM), the resulting representational bias is lower than for existing models. Moreover, CM can discover heterogeneous collaboration concepts and types such as resource sharing and message exchange, resulting in fitting and precise collaboration Petri nets. The evaluation shows that CM achieves its design goals: no assumptions on concepts and types as well as fitting and precise models, based on 26 artificial and real-world event logs from literature.

Keywords: Collaboration Mining · Collaboration Process Discovery · Inter-organizational Processes · Multi-agent Systems

1 Introduction

Process discovery aims to discover a process model from process executions recorded in an event log [8]. We distinguish two types of executed processes: *Process orchestrations* or *collaboration processes* that represent the control-flow for similar *cases* [14] or for *collaborating cases* [11] respectively. In the first case, *process orchestration discovery* (POD) techniques aim at discovering process models of process orchestrations from a set of process instances correlated by cases [14]. In the second case, *collaboration process discovery* (CPD) [31,13,21] techniques aim to discover a process model of collaboration processes from a set of process instances correlated by collaborating cases. Since collaboration processes are composed of multiple process orchestrations that jointly achieve a shared business goal, its collaborating cases contain multiple cases each corresponding to a particular process orchestration, i.e., cases and orchestrations are in a one-to-one relationship. Most of the CPD techniques target their own class of compositional Petri net to model a collaboration process, with exceptions that target *Business Process Modeling Notation*[1] (BPMN) like [17,12], but can be transformed into an equivalent Petri net. Hence, CPD techniques are characterized by targeting different compositional Petri nets, yet a standard model similar to *workflow nets* for process orchestrations is missing [11].

The decision of which model class to target in process discovery is crucial as it determines the *representational bias* [2] that implies the search space for the discovery technique. The variety among CPD techniques results from specializing on certain collaboration processes, e.g., *cross-departmental healthcare processes* (CCHP) in healthcare [21], *inter-organizational* processes [13] in various domains, and *web service compositions* [29]. These specializations justify assumptions on collaborations and their *interaction patterns* [10], e.g., only bilateral, point-to-point *message exchanges* exist for [21]. Which of the four collaboration types, i.e., message exchanges, *handover-of-work*, *resource sharing*, and *activity execution*, are supported also originates from the chosen collaboration process, e.g., only message exchanges and handover-of-work are discovered in [13]. In order to increase generalizability and lower the representational bias of current CPD, we state our research question as follows: **How can we discover fitting and precise process models of collaboration processes from a single event log in general?**

By proposing *collaboration Petri nets (cPN)* and designing the new Collaboration Miner (CM) to discover cPN, our contribution results in a generic CPD technique that mines fitting and precise collaboration process models across domains. CM discovers high quality models for all of the 22 artificial event logs that are recorded from multi-agent systems [23] and inter-organizational processes [13]. Moreover, CM discovers high quality models for the four real-world event logs that are recorded from healthcare collaboration processes [21]. As CM with its cPN target supports all four collaboration types and does not assume certain interaction patterns, the representational bias is lowered and model quality is maintained across heterogeneous collaboration processes. Note that we assume a single event log recorded from executing a collaboration process is given, i.e., we abstract from event extraction, merging, and correlation [14] with corresponding clock synchronization issues as well as privacy concerns [12].

First, basic definitions and notations are repeated in Sect. 2. The cPN formalism is introduced in Sect. 3. Section 4 presents CM by specifying event log requirements and a generic approach for discovery. An empirical evaluation of CM in comparison to existing CPD techniques is reported in Sect. 5. Next, related work is discussed in Sect. 6. Lastly, Sect. 7 concludes this paper and gives an outlook.

[1] https://www.omg.org/spec/BPMN/2.0/

2 Preliminaries

We repeat basic definitions and notations.
Let X, Y be sets.
– $\mathcal{P}(X) = \{X' \mid X' \subseteq X\}$ denotes the *powerset* of X, and $\mathcal{P}^+(X) = \mathcal{P}(X) \setminus \emptyset$ (with $\emptyset$ the empty set) denotes the *set of all non-empty subsets* of X. Given set X', the restriction of R's domain to X' is $R_{|X'} = \{(x, y) \in R \mid x \in X'\}$.
– A *trace* over X of *length* $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \to X$. For $|\sigma| = 0$, we write $\sigma = \epsilon$ and for $|\sigma| > 0$, we write $\sigma = \langle x_1, \ldots, x_n\rangle$. The *set of all finite traces* over X is denoted by X^*. We write $x \in \sigma$ for $x \in X$, if $\exists_{i \in \{1,\ldots,|\sigma|\}}\ x = \sigma(i)$.
– A *multiset (or bag)* m over X is a function $m : X \to \mathbb{N}$, i.e., $m(x) \in \mathbb{N}$ or $x^{m(x)}$ for $x \in X$ denotes the number of times x appears in m. For $x \notin X$, we define $m(x) = 0$. $\mathcal{B}(X)$ denotes *the set of all finite multisets* over X. The *support of multiset* $m \in \mathcal{B}(X)$ is defined by $\text{supp}(m) = \{x \in X \mid m(x) > 0\}$, i.e., the support is the set of distinct elements that appear in m at least once. We also write $m = [x_1^{m(x_1)}, \ldots, x_n^{m(x_n)}]$ for $\text{supp(m)} = \{x_1, \ldots, x_n\}$. Set operations (subset, addition, subtraction) are lifted to multisets in the standard way [3].
– Let Λ be a finite set of *activity labels*, where $\Lambda_{\{\tau\}} = \Lambda \cup \tau$ for $\tau \notin \Lambda$ the *silent activity* [9]. A *labelled Petri net* is a 5-tuple $N = (P, T, F, l, \Lambda_{\{\tau\}})$, where P is the set of *places*, T is the set of *transitions* with $P \cap T = \emptyset$, $F \subseteq ((P \times T) \cup (T \times P))$ is the *flow* relation, and $l : T \to \Lambda_{\{\tau\}}$ is the *transition labelling* function. We define the *preset* of $x \in P \cup T$ by $\bullet x = \{y \mid (y, x) \in F\}$ and the *postset* of x by $x\bullet = \{y \mid (x, y) \in F\}$. A multiset $m \in \mathcal{B}(P)$ is called a *marking.* Given a marking m, $m(p)$ specifies the number of tokens in place p. The *transition enabling* $(N, m)[t\rangle$ for $t \in T$ is defined by $(N, m)[t\rangle$ iff $m(p) \geq 1$ for all $p \in \bullet t$. An enabled transition $(N, m)[t\rangle$ can *fire*, denoted by $(N, m)[l(t)\rangle(N, m')$, resulting in a new marking m' defined by $m' + \bullet t = m + t\bullet$. A marking m' is *reachable* from (N, m) iff a trace of transition firings exists that starts in (N, m) and ends in (N, m'). N is a *workflow net* (WF-net) iff (i) there exists a single source place $i \in P$: $\bullet i = \emptyset$; (ii) there exists a single sink place $o \in P$: $o\bullet = \emptyset$; and (iii) every node $x \in P \cup T$ is on a directed path from i to o. The *initial marking* of N is $[i]$ and the final marking is $[o]$.

3 Collaboration Petri Nets

This section introduces collaboration Petri nets (cPN). We conceptualize the organizations/departments, agents, and services that collaborate in a collaboration process by a set of collaboration concepts $\mathcal{C}$. Each concept's dynamic behavior is a process orchestration. Hence, a *workflow collection* lists the disjunct WF-nets of each collaboration concept in a collaboration process:

Definition 1 (Workflow Collection). *Let $\mathcal{C}$ be the set of collaboration concepts in a collaboration process. A workflow collection is a tuple $WC = (N_c)_{c \in \mathcal{C}}$ of WF-nets $N_c = (P_c, T_c, F_c, l_c, \Lambda_{\{\tau\}})$ with disjunct place and transition names, i.e., $\forall_{c,c' \in \mathcal{C}}$ if $c \neq c'$, then $(P_c \cup T_c) \cap (P_{c'} \cup T_{c'}) = \emptyset$. We define:*

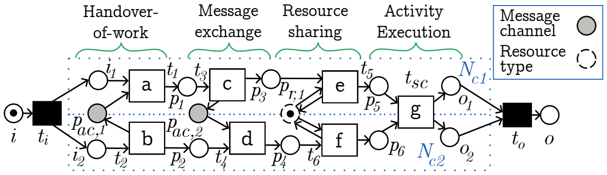


Fig. 1. Collaboration Petri net cPN with all four collaboration types.

- $T^u = \bigcup_{c \in \mathcal{C}} T_c$, $P^u = \bigcup_{c \in \mathcal{C}} P_c$, $F^u = \bigcup_{c \in \mathcal{C}} F_c$, *the sets of transitions, places, and arcs of the workflow collection respectively,*
- $l^u : T^u \to \Lambda_{\{\tau\}}$, $l^u(t) = l_c(t)$ *for* $t \in T_c$.

In Fig. 1, a cPN is depicted. Two agents "c1" and "c2" collaborate in this section's running example. $\mathcal{C}$ equals the two agent names. Each agent's process orchestration is modelled as a WF-net (denoted in blue in Fig. 1) without collaborations. We refer to the process orchestrations in a collaboration process as the "intra-process" behavior of the collaboration process.

There exist four collaboration types $\upsilon \in \Upsilon$: Message exchange (υ_m), handover-of-work (υ_h), resource sharing (υ_r), and activity execution (υ_s) [21,11]. υ_m, υ_h, and υ_r are asynchronous and υ_s is synchronous. Following existing CPD techniques [21,30], handover-of-work is a special case of message exchange. For instance, "c2" hands the work over to "c1" as represented by $p_{ac,1}$ in Fig. 1. In contrast, message exchange via asynchronous collaboration places p_{ac} can occur for any transition of a WF-net, e.g., $p_{ac,2}$. Further collaborations in Fig. 1 are: Resource sharing of resource type $p_{r,1}$ and an activity execution t_{sc} of activity "g" between "c1" and "c2". All collaborations between WF-nets of the four types are defined by a *collaboration pattern*:

Definition 2 (Collaboration Pattern). *Let* $WC = (N_c)_{c \in \mathcal{C}}$ *be a workflow collection. A collaboration pattern is a tuple* $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$*, where:*

1. P_{AC} *is the set of asynchronous collaboration places that do not intersect with existing names, i.e.,* $P_{AC} \cap (P^u \cup T^u) = \emptyset$ *(cf. Def. 1),*
2. $P_{RS} \subseteq P_{AC}$ *is the set of shared resource collaboration places,*
3. $ra : P_{RS} \to \mathbb{N}^+$ *is the resource allocation function, i.e., for shared resource type* $p_r \in P_{RS}$*, there exist* $ra(p_r)$ *shared resources,*
4. $AC = \{(p_{ac}, T_s, T_r) \in P_{AC} \times \mathcal{P}^+(T^u) \times \mathcal{P}^+(T^u) \mid \forall_{t \in T_s, t' \in T_r}\, l^u(t) \neq \tau \wedge l^u(t') \neq \tau\}$ *is the asynchronous collaboration relation, i.e.,* (p_{ac}, T_s, T_r) *with* $p_{ac} \notin P_{RS}$ *denotes that transitions* $t \in T_s$ *send a message and transitions* $t' \in T_r$ *receive a message of type* p_{ac} *via channel* p_{ac}*,*
5. *for every* $p_r \in P_{RS}$ *there exists* $(p_r, T_1, T_2) \in AC$ *such that* $T_1 = T_2$*, i.e., resource types are used and released in transitions* $t \in T_1$*, and*
6. $ET = \{(t_{sc}, T_{sc}) \in T^u \times \mathcal{P}^+(T^u) \mid t_{sc} \in T_{sc} \wedge l^u(t_{sc}) \neq \tau \wedge \forall_{t,t' \in T_{sc}}\, l^u(t) = l^u(t')\}$ *is the relation of synchronous collaborations induced by equally-labelled transitions.*

Observe that all three asynchronuous collaboration types are encoded by relation AC (4). Distinctions are made for resource sharing through P_{RS} (2), the resource allocation ra (3), and the self-loop requirement in (5). Handover-of-work is not explicitly differentiated, since its only difference is the "location" of its receiving transitions T_r within the receiving WF-nets. Formally, handover-of-work is determined by the condition: given some collaboration concept $c \in \mathcal{C}$, a transition $t \in i_c\bullet$ in the postset of source place i_c in WF-net N_c has an asynchronous collaboration place $p_{ac} \in P_{AC}$ in its preset $p_{ac} \in \bullet t$ [32]. Inducing synchronous collaboration by equally-labelled transitions (6) follows all existing CPD techniques [24,33,28,6,23,21,9] with synchronous collaboration.

Note that our collaboration pattern builds on existing techniques with respect to collaboration type modeling. The main difference is that we take a global view and we generalize the collaboration types and the message communication models of existing techniques (cf. Sect. 6). First, our definition provides a *global view* on the collaborations in a collaboration process, as all collaborations of the collaboration process are defined in a single collaboration pattern. Because the collaboration pattern is separated from the intra-process behavior of the collaboration process, our global view avoids redundancies and simplifies the discovery of collaboration processes in Sect. 4. In contrast, the *local view* of existing techniques (e.g., [21]) represents the collaborations in each concept's process orchestration, i.e., the "inter-process" collaboration behavior is included in the "intra-process" behavior. Hence, a single collaboration is included multiple times in different process orchestrations such that the collaboration has to be discovered multiple times. Second, our definition generalizes the point-to-point communication model of [12] to multiple sender and receivers per message type. Similar to [23], our model allows for different sending transitions and receiving transitions per collaboration concept and message type p_{ac}.

Given a workflow collection WC and collaboration pattern CP_{WC}, a collaboration Petri net is the result of merging the WF-nets in WC as specified by the collaboration pattern in a similar, yet generalized manner to [4].

Definition 3 (Collaboration Petri Net (cPN)). *Let $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration pattern with $WC = (N_c)_{c\in\mathcal{C}}$ a workflow collection. A Collaboration Petri Net is a marked Petri net $cPN = \biguplus^{CP}_{c\in\mathcal{C}} N_c = ((P, T, F, l, \Lambda_{\{\tau\}}), m_0)$ defined as:*

1. *$P = P^u \cup P_{AC} \cup \{i, o\}$ (cf. Def. 1),*
2. *$T = r\left(T^u\right) \cup \{t_i, t_o\}$, with r a renaming function: $r(x) = t_{sc}$ if there exists a $(t_{sc}, T_{sc}) \in ET$ such that $t \in T_{sc}$, otherwise $r(x) = x$,*
3. *$\{i, o, t_i, t_o\} \cap (P^u \cup T^u \cup P_{AC}) = \emptyset$,*
4. *$F' = F^u \cup$
$\{(t, p) \in T^u \times P_{AC} \mid (p, x, y) \in AC \wedge t \in x\} \cup$ $\{(p, t) \in P_{AC} \times T^u \mid (p, x, y) \in AC \wedge\ t \in y\} \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, i_c) \mid c \in \mathcal{C}\} \cup \{(o_c, t_o) \mid c \in \mathcal{C}\}$,*
5. *$F = \{(r(x), r(y)) \mid (x, y) \in F'\}$,*
6. *$l(t) = l^u(t)$ if $t \in T^u$, $l(t) = \tau$ otherwise,*

7. *$m_0(p) = 1$ if $p = i$, $m_0(p) = ra(p)$ if $p \in P_{RS}$ and $m_0(p) = 0$ otherwise.*

Observe that the example in Fig. 2 is a $cPN = \biguplus_{c \in \mathcal{C}}^{CP} N_c$. The collaboration pattern CP “consists of” the two asynchronous message places $p_{ac,1}, p_{ac,2}$, the asynchronous resource place $p_{r,1}$, and the synchronous activity execution transition t_{sc}. The collaboration process starts by instantiating a collaborating case as modelled by $m_0(i) = 1$. The collaborating case corresponds to a case for “c1” and to a case for “c2”. The collaborating case ends after all agent’s process orchestrations ended, i.e., the following final marking is reached $m(o) = 1$, $m(p) = ra(p)$ if $p \in P_{RS}$, and $m(p) = 0$ otherwise.

Note that resource places do not change the semantics in an untimed setting, but *cPN*s extended with time delays for transitions firings would be sensitive to resource places, i.e., discovering resource places supports subsequent analysis. Also note that the collaboration concepts in a collaboration process interact one-to-one, i.e., there exists a single instance of each collaboration concept for execution. Hence, collaboration processes only intersect with *artifact-* or *object-centric* processes [6] with respect to synchronous collaboration in a one-to-one relationship. Hence, CPD techniques cannot be generally applied in an object-centric setting. Next, we show how CM discovers *cPN*s from event logs.

4 Collaboration Miner

CM is a technique to discover a cPN from event log L. The next section introduces requirements on event logs L such that CM can be applied. In Sect. 4.2, CM with log projection π and collaboration discovery *cdisc* is defined in detail.

4.1 Event Log Requirements

CM takes an event log as input. Event logs are either generated by some collaboration process model, e.g., a cPN, or are extracted from information systems that support the process execution [14]. We apply the same conceptualization of *interleaving* semantics and totally-ordered traces of events to model business processes as the majority of discovery techniques, i.e., POD [8] and CPD [11].

Table 1. Five events (represented by rows) of real-world event log $L_{\texttt{EM}}$ [21].

Event	case	act (activity)	timestamp	c (concept)	rs (resource)	s (send_msg)	r (receive_msg)
e_1	t1	register	2019-12-28T00:20:21	{Emergency}	∅	∅	∅
e_2	t1	rescue	2019-12-28T01:20:21	{Emergency}	{charging system}	∅	
e_3	t1	reserve	2019-12-28T10:20:21	{X_ray}	{charging system}	{acceptance notice}	{reservation form}
e_4	t1	plan imaging	2019-12-28T11:20:21	{Surgical}	∅	{photo form}	{acceptance notice}
e_5	t1	consult	2019-12-28T23:20:21	{Surgical, Cardiovascular}	{diagnosis room}	∅	∅

Definition 4 (Event Log). *Let $\mathcal{A}$ and $\mathcal{V}$ be universes of attribute names and values respectively. An event is a function $e : \mathcal{A} \to \mathcal{V}$. We denote the universe of events with $\mathcal{E}$. An event log is a multiset of event traces $L \subseteq \mathcal{B}(\mathcal{E}^*)$.*

Table 1 depicts five events $e_1, \dots, e_5$ as rows with mappings $e_1(\text{case}) = \text{t1}$, $e_2(\text{act}) = \text{rescue}$, $e_3(\text{c}) = \{\text{X\_ray}\}$, $e_5(\text{rs}) = \{\text{diagnosis room}\}$, $e_4(\text{s}) = \{\text{photo form}\}$, and $e_4(\text{r}) = \{\text{acceptance notice}\}$. All five events $e_1, \dots, e_5 \in \sigma$ are in the same trace $\sigma \in L_{\texttt{EM}}$. $L_{\texttt{EM}}$ is recorded from executing a healthcare collaboration process in which hospital departments collaborate to treat patients [21].

We distinguish two requirements on event logs. Note that we assume a single event log of the collaboration process to be extracted, merged, and correlated already (cf. Sect. 1).

R1 $\forall_{\sigma \in L} \forall_{e \in \sigma}\; e(\text{act}) \neq \bot$, i.e., all events of L have a defined activity.

R2 $\forall_{\sigma \in L} \forall_{e \in \sigma}\; e(\text{c}) \subseteq \mathcal{C} \land (\exists_{e' \in \sigma} |e'(\text{c})| \geq 1 \lor (\exists_{\sigma_1, \sigma_2 \in L,\, e_1 \in \sigma_1,\, e_2 \in \sigma_2}\; e_1(\text{rs}) \cap e_2(\text{rs}) \neq \emptyset \lor e_1(\text{s}) \cap e_2(\text{r}) \neq \emptyset))$, i.e., all events in the event log L record a set of concepts in the "c" (concept) attribute and each trace records at least a synchronous collaboration, two traces share a resource, or share a message type.

POD techniques *disc* can be applied on event logs that satisfy requirement *R1*. In contrast, CM can only be applied on event logs that satisfy both requirement *R1* and *R2*. *R2* states that each event contains information on the involved collaboration concepts (attribute "c"). Additionally, each trace contains a synchronous collaboration, i.e., multiple concepts in the "c" attribute, or has at least one shared resource or message type in common with another trace. For instance, $e_4(\text{s}) = \{\text{photo form}\}$ in Tab. 1 means that during execution of activity "plan imaging" a message of type "photo form" is sent. If neither synchronous collaboration, resource sharing, nor message exchanges are recorded, the event log cannot be qualified as recording process executions from collaboration processes. We assume a first-in-first-out message channel per message type with a one-to-one relation between message types and channels similar to [33,21]. Thus, message instance identifiers are not required.

The "c" attribute enables applying CPD techniques in general, as otherwise the information on what concept has executed what activity is missing. Also, the "c" attribute enables to discover synchronous collaboration υ_s, the "rs" attributes enables discovery of resource sharing υ_r, and the "s" & "r" attributes enable discovery of message exchange υ_m and handover-of-work υ_h collaboration (cf. Sect. 3). Note that an event log recorded from a collaboration process whose collaboration concepts communicate via a *Pub/Sub* [12] communication model only meets requirements *R1* and *R2*, if the concepts communicate via messages or another collaboration type, too (cf. Sect. 7). Also, an event log L that satisfies both requirements can either be serialized into the *eXtensible Event Stream* (XES) log format [1] or into the *Object-Centric Event Log* (OCEL) format [6]. Given L, we can apply CM as proposed in the next section.

4.2 CM Algorithm

We start with introducing the log projection π to project event log L on a collaboration concept $c \in \mathcal{C}$:

Definition 5 (Log Projection). *Let $L \subseteq \mathcal{B}(\mathcal{E}^*)$ be an event log that satisfies requirements R1 and R2. Log projection on collaboration concept $c \in \mathcal{C}$ is defined by $\pi_c(L) = [\sigma_{1,c}^{L(\sigma_1)}, \ldots, \sigma_{n,c}^{L(\sigma_n)}]$ for* $\text{supp}(L) = \{\sigma_1, \ldots, \sigma_n\}$ *and $\sigma_{i,c} = \sigma_{i\,|\{e\,\in\,\mathcal{A}\,\to\,\mathcal{V}\,|\,c\,\in\,e(\text{c})\}}$*[2] *with $i \in \{1, \ldots, n\}$.*

For example, $\pi_{\text{“Emergency”}}(L_{\texttt{EM}})$ results in trace σ_1 to only contain the first two events of the five events depicted in Tab. 1. In the following, we define the Collaboration Miner (CM) and illustrate with example event log $L_{\texttt{EM}}$.

Step 1. Given event log L, the first step determines five sets and three functions by extracting attribute information from each event: The set of collaboration concepts $\mathcal{C} = \bigcup_{\sigma\,\in\,L,\;e\,\in\,\sigma}\; e(\text{c})$, the set of asynchronous message places $P_M = \bigcup_{\sigma\,\in\,L,\;e\,\in\,\sigma}\; e(\text{s}) \,\cup\, e(\text{r})$, the set of asynchronous resource sharing places $P_{RS} = \bigcup_{\sigma\,\in\,L,\;e\,\in\,\sigma}\; e(\text{rs})$, and the set of activities $\Lambda_L = \bigcup_{\sigma\,\in\,L,\;e\,\in\,\sigma} e(\text{act.})$. The function $\Lambda_s(x)$ returns the set of activities that sent message $x \in P_M$, $\Lambda_r(x)$ returns the set of activities that received message $x \in P_M$, and $\Lambda_{rs}(x)$ returns the set of activities that shared resource $x \in P_{RS}$. All three functions are determined by $\Lambda_y(x) = \bigcup_{\sigma\,\in\,L,\;e\,\in\,\sigma, e(y)=x} e(\text{act.})$ for $y \in \{\text{s, r, rs}\}$ with $x \in P_M$ if $y \neq rs$ and $x \in P_{RS}$ otherwise.

Example: For Tab. 1, we have collaboration concepts $\mathcal{C} =$\{Emergency, X-Ray, Surgical, Cardiov.\}, asynchronous message places $P_M = $ \{photo form, res. form, accept. notice, ...\}, asynchronous resource sharing places $P_{RS} =$ \{charg. system, diagn. room \}, the set of activities $\Lambda_L = $ \{ register, ...\}, the function returning sending activities per message $\Lambda_{\text{s}}(x) =$ \{(photo form, \{plan imaging\}), ...\}, the function returning receiving activities per message $\Lambda_{\text{r}}(x) =$ \{(res. form, \{reserve\}), (accept. notice, \{plan imaging\}), ...\}, and the function returning resource sharing activities per resource $\Lambda_{\text{rs}}(x) =$ \{(charg. system, \{rescue, reserve\}), ...\}.

Step 2. Project L on collection of event logs $L_{c_1}, \ldots, L_{c_n}$ with $\pi_{c_i}(L) = L_{c_i}$ for $i \in \{1, \ldots, |\mathcal{C}|\}$. Apply POD technique *disc* on each projected event log L_{c_i} resulting in a collection of WF-nets $N_{c_1}, \ldots, N_{c_n}$. Any POD technique *disc* can be applied, as long as it discovers WF-nets. Check if a valid WF-net is discovered on each projected event log. Note that if *disc* discovers *duplicate labels* [8], the respective transitions will be fused as if they represent synchronous collaboration υ_s without an additional label renaming. Construct a workflow collection $WC = (N_c)_{c \in C}$ with $N_c = (P_c, T_c, F_c, l_c, \Lambda_{L,\{\tau\}})$ by renaming place and transition names to avoid name clashes.

Example: For Tab. 1, we have $L_{c_1} = \{e_1, e_2, \ldots\}$, $L_{c_2} = \{e_3, \ldots\}$, $L_{c_3} = \{e_4, e_5, \ldots\}$, and $L_{c_4} = \{e_5, \ldots\}$. We apply *Inductive Miner* [19] as *disc*, resulting in four valid WF-nets $N_{c_1}, \ldots, N_{c_4}$ as highlighted with blue-dotted rectangles on the left in Fig. 2 (overlapping transitions t_{15}, t_{17} are to be split). Note that the place and transition names are already renamed such that $WC_{ex} = (N_c)_{c \in \mathcal{C}}$ is a workflow collection.

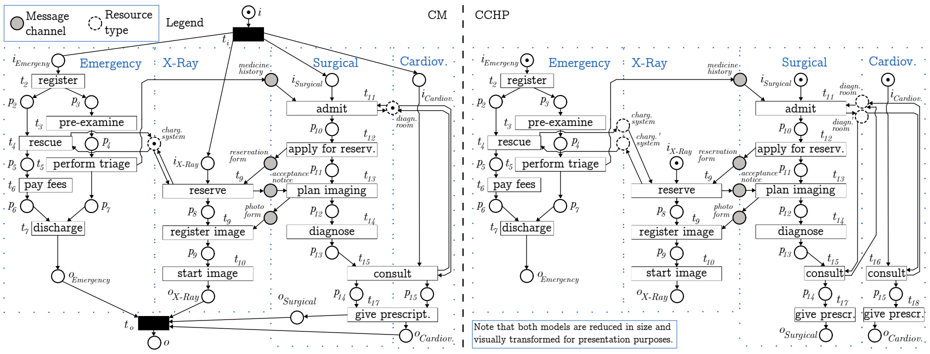
[2] We inductively define the *projection of a trace* on a set Y by $\epsilon_{|Y} = \epsilon$, $(\langle x \rangle \cdot \sigma)_{|Y} = \langle x \rangle \cdot \sigma_{|Y}$ if $x \in Y$ and $(\langle x \rangle \cdot \sigma)_{|Y} = \sigma_{|Y}$ otherwise.

Step 3. Apply collaboration discovery *cdisc* to mine collaboration pattern CP as defined in the following. Compute sending transitions $T_{\mathrm{s}}(x) = \{t \in T^u \mid l^u(t) \in \Lambda_{\mathrm{s}}(x)\}$ (cf. Def. 1), receiving transitions $T_{\mathrm{r}}(x) = \{t \in T^u \mid l^u(t) \in \Lambda_{\mathrm{r}}(x)\}$, resource sharing transitions $T_{\mathrm{rs}}(x) = \{t \in T^u \mid l^u(t) \in \Lambda_{\mathrm{rs}}(x)\}$, message exchanges $AC' = \{(p_{ac}, T_{\mathrm{s}}(p_{ac}), T_{\mathrm{r}}(p_{ac})) \mid p_{ac} \in P_M \wedge T_{\mathrm{s}}(p_{ac}) \neq \emptyset \wedge T_{\mathrm{r}}(p_{ac}) \neq \emptyset\}$, and resource sharing $AC'' = \{(p_{ac}, T_{\mathrm{rs}}(p_{ac}), T_{\mathrm{rs}}(p_{ac})) \mid p_{ac} \in P_{RS} \wedge T_{\mathrm{rs}}(p_{ac}) \neq \emptyset\}$. If events in L do not contain information on *lifecycles* [8], set $ra(p_r) = 1$ for $p_r \in P_{RS}$, else determine $\max_{p_r}(L)$ the maximum of concurrently running activities sharing p_r and set $ra(p_r) = \max_{p_r}(L)$. Then, collaboration pattern $CP = (P_{RS} \cup P_M, P_{RS}, ra, AC' \cup AC'', ET)$, where ET is induced by equally-labelled transition subsets of all transitions in WC (cf. Def. 2).

Example: For WC_{ex} (cf. Fig. 2), we have sending transitions $T_{\mathrm{s}}(x) = \{(\text{accept. not.}, \{t_9\}), \ldots\}$, receiving transitions $T_{\mathrm{r}}(x) = \{(\text{res. form}, \{t_9\}), \ldots\}$, resource sharing transitions $T_{\mathrm{rs}}(x) = \{(\text{charg. system}, \{t_4, t_9\}), \ldots\}$, message exchanges $AC' = \{(\text{res. form}, \{t_{12}\}, \{t_9\}), \ldots\}$, and resource sharing $AC'' = \{(\text{charg. system}, \{t_4, t_9\}, \{t_4, t_9\}), \ldots\}$. ra is a constant function at value 1, because $L_{\mathtt{EM}}$ does not record lifecycles. Then, $CP = (P_{RS} \cup P_M, P_{RS}, ra, AC' \cup AC'', \{(t_{15}, \{t_{15}, t_{16}\}) \ldots\})$.

Step 4. Return $cPN = ((P, T, F, l, \Lambda_{L,\{\tau\}}), m_0) = \biguplus_{c \in \mathcal{C}}^{CP} N_c$.

Example: The cPN is depicted on the left in Fig. 2.



Note that both models are reduced in size and visually transformed for presentation purposes.

Fig. 2. cPN discovered by CM and the composed RM_WF_net discovered by CCHP on log `EM`.

The parameters of CM are inherited from the respective POD technique *disc*, i.e., no new parameters are added to the algorithm. Similar to existing CPD techniques, CM applies a divide-and-conquer approach on the collaboration concept in the event log and POD technique *disc* on projected event logs (cf. **Step 2**), since a collaboration process is a composition of WF-nets. Conceptually, CM comes with a general formulation of collaboration concepts, domain-agnostic event log requirements, no assumptions on the interaction patterns for message exchanges, and supports all four collaboration types (cf. CP in **Step 3** and

Sect. 5 for details). Consequently, resulting cPNs are not specialized on certain collaboration processes.

Since CM builds on a Petri net theory with activity labels, CM supports all *disc* that discover *duplicate* ($l(t) = l(t')$) and *silent activities* ($l(t) = \tau$). In particular, silent activities are crucial for many control-flow patterns. Both activities are not fused in **Step 4** (cf. **Step 2** and Def. 2). Note that π projects to the empty trace ϵ for some $c \in \mathcal{C}$ in **Step 2**, if a trace does not contain any event with activities executed by c. The design ensures a fitting cPN, as without the ability to "skip" a WF-net corresponding traces cannot be perfectly replayed. Thus, this projection conforms to the design goals of CM.

Also, CM still returns a valid cPN in case some parts of the event logs' requirement *R2* are not satisfied: Missing "rs" attribute results in P_{RS} to be empty, missing "s" with a "r" for some message type or vice versa results in the type not to be included (cf. non-empty sets in **Step 3**), and no "s" and "r" attributes results in P_M to be empty. However, if the "c" attribute is missing, $\mathcal{C}$ is undefined and **Step 2** cannot be applied, i.e., collaboration concepts must be recorded for every event such that CM discovers a valid cPN.

4.3 Tool Support

The CM implementation is publicly available at https://gitlab.com/janikbenzin/cm in Python and builds on the PM4PY[3] library, i.e., all POD techniques that discover WF-nets in PM4PY can be applied for *disc* in CM. For the empirical evaluation in the next section, the implementation comes with an automated evaluation pipeline that supports event logs conforming to the XES extensions as defined in [23] for multi-agent systems, in [13] for BPMN collaborations, and in [21] for healthcare collaboration processes. Each of these XES event logs can be automatically converted to the respective event log format required by one of the CPD techniques in Tab. 6, e.g., XES logs are also converted to equivalent OCEL logs. CM is implemented on the XES extension as defined in [13]. From the 14 CPD techniques in Tab. 3, three publicly available CPD technique implementations are packaged with our CM implementation to facilitate reproducing the empirical evaluation automatically. While *object-centric process discovery* (OCPD) [6] is implemented in PM4PY, the CCHP [21] and *Colliery* [13] CPD techniques were originally available with a graphical user interface only. Hence, we have automated both the implementation of CCHP and Colliery such that both CPD techniques are callable via the command line.

5 Experimental Evaluation

The 15 CPD techniques depicted in Tab. 3 discover different Petri net classes or BPMN diagrams to model collaboration processes from heterogeneous domains with different collaboration types. From the 15 CPD techniques, only

[3] https://pm4py.fit.fraunhofer.de/

CCHP [20,21], Colliery [13,12], OCPD [6], and Agent Miner [30] have publicly available implementations and can be applied in our evaluation. Nevertheless, Agent Miner is excluded from the evaluation for three reasons. First, the supported collaboration type υ_h together with the assumption that only a single concept can execute an activity simultaneously (cf. Def. 4.1 in [30]) means that the Agent Miner cannot be applied to event logs with synchronous collaboration. Second, Agent Miner assumes that every directly-follows pair of events (e_1, e_2) with different concept attributes implies a handover-of-work message between the respective concepts in a trace (cf. Def. 4.2 in [30]), which is violated in every of the 26 event logs. Third, we still tried to apply Agent Miner on the logs, but were unable to get an output.

Table 2. Overview of existing CPD techniques.

CPD	Year	Model	Υ	Domains
[16]	2008	WF-nets	υ_m	Web service
[33]	2013	Integrated RM_WF_nets	$\upsilon_m, \upsilon_s, \upsilon_r$	Logistics, Healthcare
[24,28]	2013/15	Artifact-centric models	υ_s	Accounting
[29]	2019	Communication nets	υ_m	Web service
[6]	2020	Object-centric Petri nets	υ_s	Commerce
[31]	2020	Top-level process model	υ_m	Logistics
[17]	2021	BPMN Choreography	υ_m	Commerce
[15]	2022	System net	$\upsilon_s, \upsilon_m, \upsilon_h$	Retail
[18]	2022	Industry net	υ_m	Theoretical
[9]	2023	Typed Jackson nets	υ_s	Commerce
[23]	2023	Generalized WF-nets	υ_s, υ_m	Multi-agent systems
[30]	2023	Multi-agent system net	υ_h	Healthcare & other
[25]	2023	BPMN collab. diagram	υ_m, υ_h	Commerce
[20,21]	2023	Composed RM_WF_nets	Υ	Healthcare
[13,12]	2024	BPMN collab. diagram	υ_m, υ_h	Healthcare & other

Table 3. Overview of existing CPD techniques.

CPD	Year	Model	Domains
Gaaloul et al.	2008	WF-nets	Web service
Zeng & Sun et al.	2013	Integrated RM_WF_nets	Logistics, Healthcare
Nooijen et al.	2013/15	Artifact-centric models	Accounting
Stroiński et al.	2019	Communication nets	Web service
van der Aalst et al.	2020	Object-centric Petri nets	Commerce
Zeng & Duan et al.	2020	Top-level process model	Logistics
Hernandez-Resendiz et al.	2021	BPMN Choreography	Commerce
Fettke et al.	2022	System net	Retail
Kwantes et al.	2022	Industry net	Theoretical
Barenholz et al.	2023	Typed Jackson nets	Commerce
Nesterov et al.	2023	Generalized WF-nets	Multi-agent systems
Tour et al.	2023	Multi-agent system net	Healthcare & other
Peña t al.	2023	BPMN collab. diagram	Commerce
Liu et al.	2023	Composed RM_WF_nets	Healthcare
Corradini et al.	2024	BPMN collab. diagram	Healthcare & other

As generation/extraction is out of scope for this paper, event logs from literature are selected. The selection criterion of meeting requirements *R1* and *R2* yields 26 event logs[4]: 12 artificial event logs from multi-agent systems [22,23], 10 artificial event logs from BPMN collaborations [13], and 4 real-world event logs from healthcare collaboration processes [21]. For 17/22 artificial event logs (1-A5 in Tab. 6), the *true* process models that generated the respective event log are available or convertible by PM4PY's "BPMN to Petri net" (see True in Tab. 6). For the five artificial event logs R1-R5 conversion with PM4PY is not possible due to the BPMN model structure, so their true process models are not available. Also, true process models are not available for real-world event logs. Descriptive statistics of the 26 event logs are reported in Tab. 6. The event logs vary along the dimensions: # of events, average trace length, # of collaboration concepts, collaboration types υ, and properties describing the respective interaction pattern of message exchanges υ_m [10]. Interaction patterns vary along the maximum number of collaboration concepts interacting through a message type, i.e., *one*-way and *two*-way bilateral or *multi*lateral interactions, the maximum number of transmissions per type, i.e., *single* or *multi*, and the relation between activities sending/receiving messages per type, i.e., point-to-point (denoted by 1:1 in Tab. 6), one-to-many (1:n), and many-to-many interaction (m:n).

Each of the 26 event logs is converted to the respective CPD technique's event log input format. We apply CM, CCHP, Colliery, and OCPD on each converted event log with the Inductive Miner [19] for POD *disc* to ensure result differences being caused by the CPD techniques properties, e.g., supported collaboration types, and design choices, e.g., assumptions on the interaction pattern. Inductive Miner is chosen for its formal guarantee to discover perfectly fitting WF-nets such that this design goal can be achieved. We report the model *size* as the sum of the # of places and # of transitions in Tab. 6. We apply *alignment-based fitness* and *precision* [5,7] to measure model quality with PM4PY except for logs ID to SD in Tab. 6 for which we manually computed fitness and precision (annotated with *) using the more efficient ProM plugin with similar parameters, as PM4PY exceeds a space limit of >58GB on a Fedora machine with 14-core Intel i5-13500T (13th Gen) CPU and 64 GB RAM.

[26,27] propose monotone alternatives to both alignment-based metrics. However, neither version of the monotone alternatives is suitable for our evaluation. First, the perfectly-fitting version results in metrics equal to zero as soon as no trace in the event log can be replayed by the discovered model. If a CPD technique does not support synchronous collaboration υ_s, e.g., Colliery, it discovers a model with duplicate labels for synchronous collaboration υ_s due to projection. As logs may contain only traces with υ_s, metrics are zero and too low. For example, 7/8 logs with υ_s contain only traces with υ_s (9-11 and EM-SD in Tab. 6). Second, the partially-fitting version often exceeds a heap space limit of 58GB (>3x the space of [8]). Third, alignments allow arbitrary final markings. In contrast, the monotone metrics do not take the final marking as

[4] Note that the smart agriculture event log in [12] only contains signalling between concepts (Pub/Sub) and does not meet the requirements (cf. Sect. 4.1).

input[5], but compute it based on the assumption that places in the final marking are sink places. CCHP requires a final marking for resource places that are neither a sink place nor self-loops [21], so resource places cannot be removed (cf. Sect. 3) and monotone metrics cannot be computed for CCHP models with resource sharing. Altogether, alignment-based metrics are more suitable, as they can handle partially-fitting traces, are more efficient in terms of heap space, and more flexible in terms of final markings.

We illustrate differences of CM and CCHP with their Petri nets discovered on $L_{\mathtt{EM}}$. CCHP is "closest" to CM, as CCHP similarly claims to discover all four collaboration types (cf. Tab. 3) and applies a comparable approach to discovery. Hence, the illustration supports understanding the following different results. In Fig. 2, the cPN discovered by CM (CM's model) and on the right the composed RM_WF_net (CCHP's model) is depicted. While CM's model has a global source and sink place, CCHP does not. Hence, CM's model has the notion that a token in i represents a collaborating case. Most parts of the two models are similar, as CCHP is proposed using $L_{\mathtt{EM}}$, i.e., $L_{\mathtt{EM}}$ does not violate any assumptions by CCHP. CCHP's model has two problems. First, it does not discover υ_s as claimed, since it discovers duplicate activities for "consult" and "give prescription". Second, it does not discover self-loop places for resources and no resource allocation such that "Surgical" and "Cardiov." can never reach their sink place. With marked resource places, CCHP's model would force either "Surgical" or "Cardiov." to execute "consult".

Considering fitness, CM is the only CPD technique that always discovers a cPN that perfectly fits event log L (cf. Tab. 6). CCHP discovers Petri nets whose final markings are unreachable for event logs `1`, `2`, and `4-11`, as CCHP implicitly assumes a point-to-point (1:1) activity relation, but still adds an interaction for each unique activity pair, e.g., a 2:2 relation yields four activity pairs. As resource places are not discovered as self-loop places by CCHP (cf. Fig. 2), the only log `EM` with resource sharing results in a Petri net with unreachable final marking. Since support for synchronous collaboration υ_s is not implemented in CCHP, Petri nets discovered on event logs with υ_s have a lower fitness similar to Colliery that does not support υ_s by design. Because OCPD only supports υ_s (cf. Tab. 3), discovered Petri nets correspond to the parallel execution of all collaboration concept WF-nets for event logs without υ_s. If υ_s is contained in an event log, the parallel branches that correspond to a collaboration concept WF-net are synchronized by synchronous collaborations. Instead of projecting logs, OCPD *flattens* [6] the log such that the empty trace ϵ can never be in a flattened event log. Consequently, a concept's WF-net can never be skipped, but skipping is required for logs `A5`, `R3`, and `R5`. Therefore, fitness is not perfect for these logs. Also, CCHP and Colliery do not allow ϵ in projected event logs such that CM is the only technique that achieves perfect fitness on logs `A5`, `R3`, and `R5`.

Considering precision, CM and Colliery discover the most precise models for 15/26 and 12/26 event logs respectively, i.e., either CM or Colliery discover the

[5] https://github.com/jbpt/codebase/tree/master/jbpt-pm/entropia

Table 4. Model quality metrics of the true model, if available, and discovered by CM, CCHP [21], Colliery [13], and OCPD [6] based on artificial event logs 1-12 [22,23], A1-A5 & R1-R5 [13], and real-world event logs EM-SD [21], where $\Upsilon_\sigma = \{\upsilon_x \mid x \in \sigma\}$.

Event log L		1	2	3	4	5	6	7	8	9	10	11	12	A1
# Events		95052	149988	92668	102404	182452	123322	88068	157098	115000	102548	160000	88089	100
Avg. trace length		19	30	19	20	36	25	18	31	23	21	32	18	8
# Col. concepts		2	2	2	2	2	2	2	3	2	2	2	2	2
Col. types υ		υ_m	υ_m	υ_m	υ_m	υ_m	υ_m	υ_m	υ_m	υ_m, υ_s	υ_m, υ_s	υ_m, υ_s	υ_m, υ_s	υ_m
υ_m: Max. col. con.		one	one	one	two	two	two	two	multi	two	two	two	two	one
υ_m: Max. trans.		single	single	single	single	single	single	multi	multi	single	single	single	single	single
υ_m: Activity rel.		1:n	m:n	1:1	m:n	1:n	1:n	m:n	m:n	m:n	m:n	1:n	1:n	1:1
True	Precision	0.716	0.401	0.754	0.759	0.39	0.564	0.817	0.481	0.714	0.793	0.495	0.766	0.972
	Size	66	100	88	76	109	113	61	128	105	78	94	86	22
CM	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.736	0.351	**0.765**	**0.792**	0.208	0.586	**0.817**	**0.504**	**0.716**	**0.781**	0.433	**0.758**	**0.972**
	Size	75	125	103	92	167	132	77	148	120	88	125	95	26
CCHP	Fitness	ex	ex	**1.0**	ex	ex	ex	ex	ex	ex	ex	ex	0.384	**1.0**
	Precision	ex	ex	**0.765**	ex	ex	ex	ex	ex	ex	ex	ex	0.583	**0.972**
	Size	78	127	**95**	92	165	135	78	150	121	94	128	99	26
Colliery	Fitness	0.867	0.966	**1.0**	0.71	0.964	0.924	0.667	0.699	0.641	0.707	0.93	0.893	**1.0**
	Precision	**0.738**	**0.426**	**0.765**	0.747	**0.26**	**0.598**	0.686	0.306	0.5	0.67	**0.465**	0.697	**0.972**
	Size	79	131	103	94	169	136	77	295	130	96	128	104	26
OCPD	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.714	0.341	0.719	0.701	0.136	0.545	0.594	0.387	0.677	0.726	0.185	0.717	0.753
	Size	**74**	**123**	101	**90**	**121**	**128**	**74**	**142**	**118**	**86**	**106**	**93**	**24**

Event log L		A2	A3	A4	A5	R1	R2	R3	R4	R5	EM	ID	FP	SD
# Events		100	100	100	100	22	100	100	100	100	18909	50427	37816	4320
Avg. trace length		18	23	6	24	7	15	18	18	13	32	25	25	23
# Col. concepts		2	3	2	4	2	2	3	3	3	6	6	4	4
Col. types υ		υ_m	$\Upsilon_{\langle m,h \rangle}$	υ_m	$\Upsilon_{\langle m,h \rangle}$	υ_m	υ_m	$\Upsilon_{\langle m,h \rangle}$	$\Upsilon_{\langle m,h \rangle}$	$\Upsilon_{\langle m,h \rangle}$	Υ	$\Upsilon_{\langle m,s,h \rangle}$	$\Upsilon_{\langle m,s,h \rangle}$	$\Upsilon_{\langle m,s,h \rangle}$
υ_m: Max. col. con.		two	one	two	two	two	two	two	two	two	two	two	two	two
υ_m: Max. trans.		single	single	single	single	single	multi	multi	single	single	single	single	single	single
υ_m: Activity rel.		1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1
True	Precision	0.648	0.645	0.998	0.646	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	Size	47	59	18	63	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
CM	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.606	0.435	**0.998**	0.44	**0.538**	0.419	0.293	0.649	**0.945**	**0.986**	**0.797***	**0.789***	**0.845***
	Size	50	64	22	75	20	42	80	54	71	95	74	74	65
CCHP	Fitness	0.935	0.958	**1.0**	0.905	**1.0**	0.894	0.974	0.889	0.844	ex	**1.0**	0.943	0.979
	Precision	**0.648**	0.467	**0.998**	0.471	**0.538**	0.422	**0.369**	**0.758**	**0.945**	ex	**0.797***	0.615*	0.777*
	Size	51	65	22	71	20	41	84	55	61	105	74	76	66
Colliery	Fitness	0.935	0.935	**1.0**	0.888	0.6	0.894	0.846	0.757	0.785	0.918	0.865	0.88	0.857
	Precision	**0.648**	**0.645**	**0.998**	**0.646**	0.333	**0.439**	0.333	0.638	0.814	0.253	0.233	0.176	0.229
	Size	51	63	22	67	**12**	43	82	48	61	81	**62**	67	**51**
OCPD	Fitness	**1.0**	**1.0**	**1.0**	0.971	**1.0**	**1.0**	0.995	**1.0**	0.868	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.598	0.399	0.703	0.314	**0.538**	0.253	0.193	0.201	0.251	0.148	0.409*	0.530*	0.509*
	Size	**48**	**60**	**20**	**65**	20	**30**	**66**	**41**	**59**	**76**	68	**66**	58

Table 5. Model quality metrics of the true model, if available, and discovered by CM, CCHP [21], Colliery [13], and OCPD [6] based on artificial event logs 1-12 [22,23], A1-A5 & R1-R5 [13], and real-world event logs EM-SD [21], where $\Upsilon_\sigma = \{\upsilon_x \mid x \in \sigma\}$.

Event log L		1	2	3	4	5	6	7	8	9	10	11	12	A1
# Events		95.052	149.988	92.668	102.404	182.452	123.322	88.068	157.098	115.000	102.548	160.000	88.089	100
# Col. concepts		2	2	2	2	2	2	2	3	2	2	2	2	2
Col. types		msg	msg	msg	msg	msg	msg	msg	msg	msg, sync	msg, sync	msg, sync	msg, sync	msg
υ_m: Activity rel.		1:n	m:n	1:1	m:n	1:n	1:n	m:n	m:n	m:n	m:n	1:n	1:n	1:1
True	Precision	0.716	0.401	0.754	0.759	0.39	0.564	0.817	0.481	0.714	0.793	0.495	0.766	0.972
	Size	66	100	88	76	109	113	61	128	105	78	94	86	22
CM	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.736	0.351	**0.765**	**0.792**	0.208	0.586	**0.817**	**0.504**	**0.716**	**0.781**	0.433	**0.758**	**0.972**
	Size	75	125	103	92	167	132	77	148	120	88	125	95	26
CCHP	Fitness	ex	ex	**1.0**	ex	ex	ex	ex	ex	ex	ex	ex	0.384	**1.0**
	Precision	ex	ex	**0.765**	ex	ex	ex	ex	ex	ex	ex	ex	0.583	**0.972**
	Size	78	127	**95**	92	165	135	78	150	121	94	128	99	26
Colliery	Fitness	0.867	0.966	**1.0**	0.71	0.964	0.924	0.667	0.699	0.641	0.707	0.93	0.893	**1.0**
	Precision	**0.738**	**0.426**	**0.765**	0.747	**0.26**	**0.598**	0.686	0.306	0.5	0.67	**0.465**	0.697	**0.972**
	Size	79	131	103	94	169	136	77	295	130	96	128	104	26
OCPD	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.714	0.341	0.719	0.701	0.136	0.545	0.594	0.387	0.677	0.726	0.185	0.717	0.753
	Size	**74**	**123**	101	**90**	**121**	**128**	**74**	**142**	**118**	**86**	**106**	**93**	**24**

Table 6. Model quality metrics of the true model, if available, and discovered by CM, CCHP [21], Colliery [13], and OCPD [6] based on artificial event logs 1-12 [22,23], A1-A5 & R1-R5 [13], and real-world event logs EM-SD [21], where $\Upsilon_\sigma = \{\upsilon_x \mid x \in \sigma\}$.

Event log L		A2	A3	A4	A5	R1	R2	R3	R4	R5	EM	ID	FP	SD
# Events		100	100	100	100	22	100	100	100	100	18.909	50.427	37.816	4.320
# Col. concepts		2	3	2	4	2	2	3	3	3	6	6	4	4
Col. types		msg	msg, ho	msg	msg, ho	msg	msg	msg, ho	msg, ho	msg, ho	msg, ho, sync, res	msg, ho, sync	msg, ho, sync	msg, ho, sync
υ_m: Activity rel.		1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1	1:1
True	Precision	0.648	0.645	0.998	0.646	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	Size	47	59	18	63	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
CM	Fitness	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.606	0.435	**0.998**	0.44	**0.538**	0.419	0.293	0.649	**0.945**	**0.986**	**0.797***	**0.789***	**0.845***
	Size	50	64	22	75	20	42	80	54	71	95	74	74	65
CCHP	Fitness	0.935	0.958	**1.0**	0.905	**1.0**	0.894	0.974	0.889	0.844	ex	**1.0**	0.943	0.979
	Precision	**0.648**	0.467	**0.998**	0.471	**0.538**	0.422	**0.369**	**0.758**	**0.945**	ex	**0.797***	0.615*	0.777*
	Size	51	65	22	71	20	41	84	55	61	105	74	76	66
Colliery	Fitness	0.935	0.935	**1.0**	0.888	0.6	0.894	0.846	0.757	0.785	0.918	0.865	0.88	0.857
	Precision	**0.648**	**0.645**	**0.998**	**0.646**	0.333	**0.439**	0.333	0.638	0.814	0.253	0.233	0.176	0.229
	Size	51	63	22	67	**12**	43	82	48	61	81	**62**	67	**51**
OCPD	Fitness	**1.0**	**1.0**	**1.0**	0.971	**1.0**	**1.0**	0.995	**1.0**	0.868	**1.0**	**1.0**	**1.0**	**1.0**
	Precision	0.598	0.399	0.703	0.314	**0.538**	0.253	0.193	0.201	0.251	0.148	0.409*	0.530*	0.509*
	Size	**48**	**60**	**20**	**65**	20	**30**	**66**	**41**	**59**	**76**	68	**66**	58

most precise model with the exception of CCHP for R3 and R4. In particular, the most precise model is in the same range as the true model's precision. CM can discover most precise models across all three different event log groups and, thus, across multi-agent systems, inter-organizational processes and healthcare collaboration processes. In particular, CM discovers most precise models for the four real-world event logs EM-SD that contain all or the majority of collaboration types. Colliery does not discover precise Petri nets for the four real-world event logs, because these event logs record many collaborations that are not supported by Colliery. Considering subpar precision metrics of CM, results are often close to the best precision, e.g., event log 1 with 0.736 vs. 0.738 or 5 with 0.586 vs. 0.598 . CCHP discovers the most precise model for 9/26 event logs due to the final marking being unreachable for 11/26 event logs. OCPD usually discovers the least precise model due to the low precision of parallelly executed WF-nets without any message exchange.

Considering size, OCPD regularly discovers the smallest model, since it does not discover asynchronous places. CCHP, Colliery, and CM are typically in the same range of size. An exception is the event log 1 for which Colliery discovers a model of twice the size discovered by other CPD techniques. Overall, the sizes of discovered models are close to each other and usually in the range of 1.2x the true model size.

To sum up, the results show that support for synchronous collaboration increases fitness, support for asynchronous collaboration increases precision, and violated assumptions on interaction patterns significantly decrease fitness and precision and can lead to models in which the final marking is not reachable. The experimental evaluation with 26 artificial and real-world event logs with a diverse set of collaboration processes, collaboration types, and interaction patterns shows that CM achieves its design goals of precise and fitting process models without assumptions on concepts, types or patterns.

6 Related Work

To start with, we elaborate on our differences in detail to CCHP [20,21] that is the CPD technique closest to CM. CCHP [20,21] shows multiple inconsistencies between paper and implementation; hence we use CCHP sources https://github.com/promworkbench/ShandongPM/ as substitute for parts that are undefined, e.g., *cdisc*. While CCHP has a similar divide-and-conquer approach and modelling of collaboration types, it differs in several aspects in which CM improves CCHP as shown in Sect. 5. CCHP does not allow empty traces in projected logs, which lead to reduced fitness. It assumes a one-to-one activity relation per message channel, resulting in unreachable final markings. v_s is only theoretically discovered and v_r is practically not discovered as self-loop places. CCHP does not discover resource allocations and does not specify event log requirements, which leaves an early decision of applicability to the user. Lastly, CCHP is not defined with activity labels such that duplicate and silent activities are always fused. Hence, POD techniques such as the Inductive Miner lead to undesirable results. Overall, CM generalizes, improves, and extends CCHP.

Regarding collaboration types, this work provides a global view on the collaboration process to reduce redundancies and separates the process orchestrations (intra-process) from the collaborations (inter-process) to simplify the discovery of collaboration processes. Not separating intra-process from inter-process behavior as done in the RM_WF_nets [33] requires to simultaneously discover both behaviors for each collaboration concept. As collaborations are nonetheless discovered, collaborations are discovered multiple times.

In the following, we give a quick overview on a selection of the remaining 14 CPD techniques (cf. Tab. 3). [16] discover web service compositions by conceptualizing each message exchange as a single activity. [33] discover inter-organizational processes without choices. [29] propose to discover hierarchically structured services that collaborate via message exchanges. [31] discover *top-level process models* in which the inter-process level is discovered and, subsequently, refined with *local* process models. [17] discover process choreographies with a focus on issues during merging of distributed event logs. [12,13] discover BPMN collaboration diagrams that are the result of converting discovered WF-nets N_c for each partner to BPMN and connecting activities with *message flows*. [12] is the only CPD technique that supports a *Pub/Sub* communication model in which the collaboration concepts communicate via *signals* and a sent signal can be received multiple times by different concepts. [18] formally analyse the extent to which POD techniques *disc* can be applied to discover models of asynchronously collaborating systems. [23] propose to discover a *generalized* WF-net (WF-nets that have multiple source and sink places) for each agent that collaborate via message exchanges and synchronous activity execution with each other using a soundness-guaranteeing PD technique *disc* such as the Inductive Miner [19]. Next, [30] propose to discover a *multi-agent system net* (MAS) net, which is a WF-net with labels that include the activity label and the agent executing the activity. The agents in a MAS net collaborate via handover-of-work. In general, CM advances CPD techniques in the direction of a generalized formulation, fewer assumptions, discovery of resource allocations, support of all four collaboration types, of silent activities, and of more POD techniques *disc*.

Due to CM's goal to discover models of collaboration processes and their one-to-one correspondence between process instances of collaboration concepts, it cannot be equally applied to event logs with one-to-many or many-to-many relationship between process instances of collaboration concepts or more specifically object types. Consequently, CM is only related to CPD techniques [24,28,6,15,9] on event logs satisfying requirements *R1*, *R2*, and no multiplicities between collaboration concept identifiers exist. These CPD techniques aim to integrate the control-flow and data perspective and have, thus, a different goal. Due to the intersection on certain event logs, we still subsume [24,28,6,15,9] under CPD techniques in this paper despite their different goal. [24,28] discover *artifact-centric* process models consisting of multiple artifact lifecycle models discovered from Enterprise Resource Planning systems. [15] delineate a discovery framework with *true concurrent* semantics that does not require a total order on events in the event log. [9] propose a framework for (re-)discovering *typed Jackson nets* that are a block-structured, sound-by-design subclass of *typed Petri nets with identifiers*.

7 Conclusion and Outlook

We propose Collaboration Miner (CM) to discover fitting and precise process models of collaboration processes. Among the heterogeneous set of existing target model classes proposed to model collaboration processes, CM proposes collaboration Petri nets (cPN). The cPN target class along with the CM's divide-and-conquer approach on the event log and custom collaboration discovery *cdisc* lowers the representational bias between discovered models and the true collaboration process. More specifically, CM generalizes over domains and their collaboration processes through collaboration concepts, types, and event log requirements. In addition, *cdisc* eliminates assumptions of existing techniques on the interaction patterns in the event log. CM's ability to discover high-quality models across domains and interaction patterns is empirically shown on 26 artificial and real-world event logs. Future directions are towards providing soundness guarantees, discovering multiplicities between collaboration concept instances, and supporting the Pub/Sub communication model for collaboration type υ_m. Extending CM to also discover process models that can represent multiplicities between collaboration concept instances necessitates an extension to cPNs that would bring collaboration and object-centric process mining closer together.

References

1. eXtensible Event Stream (XES). IEEE Std 1849-2016 pp. 1–50 (Nov 2016)
2. van der Aalst, W.M.P.: On the Representational Bias in Process Mining. In: 2011 IEEE WETICE. pp. 2–7 (Jun 2011)
3. van der Aalst, W.M.P., et al.: Soundness of workflow nets: classification, decidability, and analysis. Form. Asp. Comput. **23**(3), 333–363 (2011)
4. van der Aalst, W.M.P.: Modeling and analyzing interorganizational workflows. In: Proceedings 1998 ACSD. pp. 262–272 (1998)
5. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining and Knowledge Discovery **2**(2), 182–192 (2012)
6. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundamenta informaticae **175**(1-4), 1–40 (2020)
7. Adriansyah, A., Munoz-Gama, J., Carmona, J., Van Dongen, B.F., Van Der Aalst, W.M.: Measuring precision of modeled behavior. Inf Syst E-Bus Manag **13**(1), 37–67 (2015)
8. Augusto, A., et al.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Trans Knowl Data Eng **31**(4), 686–705 (2019)
9. Barenholz, D., Montali, M., Polyvyanyy, A., Reijers et al., H.A.: There and Back Again. In: PETRI NETS 2023. pp. 37–58 (2023)

10. Barros, A., Dumas, M., ter Hofstede, A.H.M.: Service Interaction Patterns. In: BPM. pp. 302–318 (2005)
11. Benzin, J.V., Rinderle-Ma, S.: Petri Net Classes for Collaboration Mining: Assessment and Design Guidelines. In: Process Mining Workshops (2024)
12. Corradini, F., Pettinari, S., Re, B., Rossi, L., Tiezzi, F.: A technique for discovering BPMN collaboration diagrams. SoSyM (2024)
13. Corradini, F., Re, B., Rossi, L., Tiezzi, F.: A Technique for Collaboration Discovery. In: Enterprise, Business-Process and Inf, Syst. Modeling. pp. 63–78 (2022)
14. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. Data Mining and Knowl. Disc. **10**(3) (2020)
15. Fettke, P., Reisig, W.: Systems Mining with Heraklit: The Next Step (2022), arXiv:2202.01289 [cs]
16. Gaaloul, W., Baïna, K., Godart, C.: Log-based mining techniques applied to Web service composition reengineering. Serv. Oriented Comp. Appl. **2**(2), 93–110 (2008)
17. Hernandez-Resendiz, J.D., Tello-Leal, E., Marin-Castro, H.M., Ramirez-Alcocer, U.M., Mata-Torres, J.A.: Merging Event Logs for Inter-organizational Process Mining. In: New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques, pp. 3–26. Springer (2021)
18. Kwantes, P., Kleijn, J.: Distributed Synthesis of Asynchronously Communicating Distributed Process Models. In: ToPNoC, pp. 49–72 (2022)
19. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: PETRI NETS 2013. pp. 311–329 (2013)
20. Liu, C., Li, H., Zeng, Q., Lu et al., T.: Cross-Organization Emergency Response Process Mining: An Approach Based on Petri Nets. Math. Probl. Eng. **2020**, e8836007 (2020)
21. Liu, C., Li, H., Zhang, S., Cheng, L., Zeng, Q.: Cross-Department Collaborative Healthcare Process Model Discovery From Event Logs. IEEE Trans. Autom. Sci. Eng. **20**(3), 2115–2125 (2023)
22. Nesterov, R.: Compositional discovery of architecture-aware and sound process models from event logs of multi-agent systems: experimental data. (May 2021)
23. Nesterov, R., Bernardinello, L., Lomazova, I., Pomello, L.: Discovering architecture-aware and sound process models of multi-agent systems: a compositional approach. SoSyM (1), 351–375 (2023)
24. Nooijen, E.H.J., van Dongen, B.F., Fahland, D.: Automatic Discovery of Data-Centric and Artifact-Centric Processes. In: BPM Workshops. pp. 316–327 (2013)
25. Peña, L., Andrade, D., Delgado, A., Calegari, D.: An Approach for Discovering Inter-organizational Collaborative Business Processes in BPMN 2.0. In: PM Workshops. pp. 487–498. Springer (2024)
26. Polyvyanyy, A., Kalenkova, A.: Monotone Conformance Checking for Partially Matching Designed and Observed Processes. In: ICPM. pp. 81–88 (2019)
27. Polyvyanyy, A., Solti, A., Weidlich, M., Ciccio et al., C.D.: Monotone Precision and Recall Measures for Comparing Executions and Specifications of Dynamic Systems. ACM Trans. Softw. Eng. Methodol. **29**(3), 17:1–17:41 (2020)
28. Popova, V., Fahland, D., Dumas, M.: Artifact Lifecycle Discovery. Int. J. Coop. Inf. Syst. **24**(01), 1550001 (2015)
29. Stroiński, A., Dwornikowski, D., Brzeziński, J.: A Distributed Discovery of Communicating Resource Systems Models. Trans. Serv. Comput. **12**(2), 172–185 (2019)

30. Tour, A., Polyvyanyy, A., Kalenkova, A., Senderovich, A.: Agent Miner: An Algorithm for Discovering Agent Systems from Event Data. In: BPM. pp. 284–302 (2023)
31. Zeng, Q., Duan, H., Liu, C.: Top-Down Process Mining From Multi-Source Running Logs Based on Refinement of Petri Nets. IEEE Access **8**, 61355–61369 (2020)
32. Zeng, Q., Lu, F., Liu, C., Duan et al., H.: Modeling and Verification for Cross-Department Collaborative Business Processes Using Extended Petri Nets. IEEE Trans. Syst. Man Cybern.: Syst. **45**(2), 349–362 (Feb 2015)
33. Zeng, Q., Sun, S., Duan, H., Liu et al., C.: Cross-organizational collaborative workflow mining from a multi-source log. Decis Support Syst **54**, 1280–1301 (Feb 2013)