Collaboration Petri Nets: Verification, Equivalence, and Discovery (Extended Version)

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. Process modeling and discovery techniques aim to construct sound and valid process models for different types of processes, i.e., process orchestrations and collaboration processes. Orchestrations represent behavior of cases within one process. Collaboration processes represent behavior of collaborating cases within multiple process orchestrations that interact via collaboration concepts such as organizations, agents, objects, and services. The heterogeneity of collaboration concepts and types such as message exchange and resource sharing has led to different representations and discovery techniques for collaboration process models, but a standard model class is lacking. We propose collaboration Petri nets (cPN) to achieve comparability between techniques, to enable approach and property transfer, and to build a standardized collaboration mining pipeline similar to process mining. For cPN, we require desirable modeling power, decision power, modeling convenience, and relations to existing model classes. We show the representation of collaboration types, structural characterization as workflow nets, automatic verification of soundness, bisimulation equivalence to existing model classes, and application in a general discovery framework. As empirical evidence to discover cPN, we conduct a comparative evaluation between three discovery techniques on a set of existing collaboration event logs.

Keywords: Collaboration Mining · Collaboration Process Models · Standardisation of Nets · Process Discovery

1 Introduction

Business processes are vital to many domains such as healthcare and manufacturing [42] and define the control-flow of business activities, i.e., what work has to be done in what order [48]. Languages for capturing process logic in process models [45,36] and techniques to discover (business) process models from process executions recorded in an event log [10,13] are are both essential to document, understand, and improve processes [25] such that the business benefits significantly [46]. Among the set of process modeling languages, Petri nets are well-suited due to their formal semantics, graphical nature, expressiveness, analysis techniques, and tool support [61,6,64]. Moreover, different modeling languages can be transformed to Petri nets [64] and metrics measuring the quality

of discovered models with respect to the event log are only defined for Petri nets [13], which means that the theory of Petri nets is central to process modeling and discovery.

Depending on the type of process, modeling and discovery techniques employ distinct approaches. *Process orchestrations* define the control-flow for similar *cases* [48,10]. Process mining techniques [42] mostly deal with process orchestrations, i.e., process discovery [13] aims at discovering a Petri net from a set of process instances correlated by cases [30] to model a process orchestration. *Collaboration processes* define the control-flow for similar *collaborating cases* [17]. As a collaboration process consists of multiple process orchestrations that collaborate to achieve a common business goal, its collaborating cases consist of multiple cases each corresponding to one of the process orchestrations. Hence, *collaboration process discovery* (CPD) techniques such as [15,24,53] aim at discovering a Petri net from a set of process instances grouped by collaborating cases. A divide-and-conquer approach on the case notion is typical for CPD techniques to apply process discovery on projected event logs.

Collaboration between cases can be heterogeneous [17] and occurs via various types of *collaboration concepts* such as a hospital's departments [53], a company's agents [72], a shop's objects like orders or packages [8], or the services in a *web service composition* [69]. Hence, a collaboration process consists of multiple collaboration concepts whose intra-process behavior is modeled as a process orchestration, while their collaborations constitute the inter-process behavior. Although different proposals exist to model and discover collaboration processes in various domains, e.g., *composed RM_WF_nets* for healthcare [53], *object-centric Petri nets* for commerce [8], and *interaction Petri nets* for supply-chains and logistics [77,29,37], a standard Petri net class is missing [17]. Similar to the de-facto standard of *workflow nets* to model process orchestrations in process mining, a standard for collaboration processes achieves comparability between techniques, enables transfer of approaches and properties, and lays the foundation for a standardized and modular collaboration mining pipeline.

Advances in collaboration mining are recently focused on CPD [53,15,72]. Following this, the requirements for a standard are: (i) sufficient *modeling power* to represent the control-flow of collaboration processes across domains and types of collaboration concepts, balanced with (ii) enough *decision power* to decide relevant problems like *soundness*, (iii) adequate *modeling convenience* such that models are relatively simple, concise, and particularly suitable for discovery, and (iv) desirable relations to existing model classes. Hence, our research question is: **How can we define a standard Petri net class for collaboration process discovery that meets the requirements for a standard?**

We propose *collaboration Petri nets (cPN)* as a standard to model and discover collaboration processes. To illustrate the proposal, Fig. 1 depicts a hospital's patient treatment collaboration process [77,76,53] modeled as a *cPN*. Collaboration concepts comprise departments "emergency", "X-ray", "surgical", and "cardiovascular". Patients are initially treated in "emergency", which collaborates via the shared resource type "charging system" (see $p_{r,1}$ in Fig. 1) with

"X-ray" and a *handover-of-work* message over *message channel* $p_{ac,1}$ with "cardiovascular". Other collaborations (inter-process) are a second *resource sharing* of the two diagnosis rooms $p_{r,2}$, *message exchanges* via channels $p_{ac,2}, \ldots, p_{ac,7}$, and synchronous *activity execution* in transitions $t_{sc,1}$ and $t_{sc,2}$ where doctors of "surgical" and "cardiovascular" consult and prescribe together. By focusing on the intra-process behavior of the four departments, four process orchestrations modeled as workflow nets are visible.

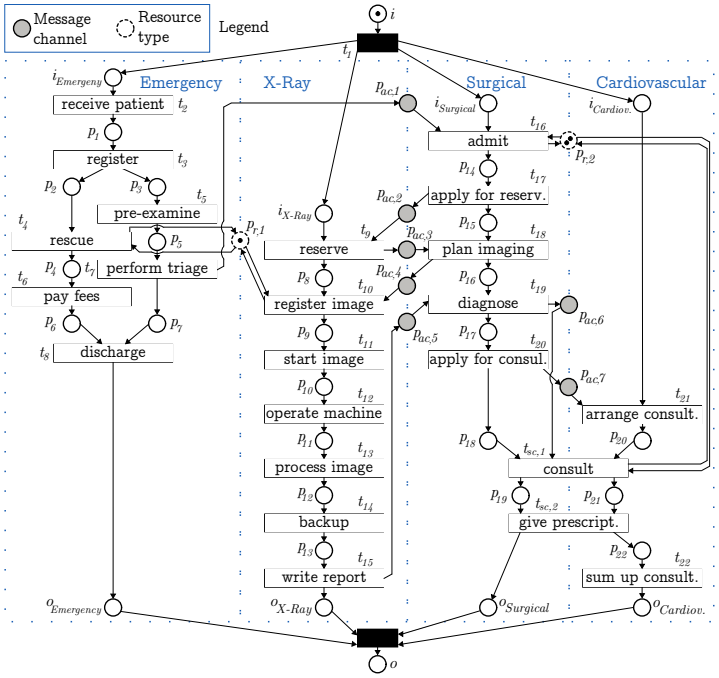

Fig. 1. *cPN* of treatment collaboration process (simplified from [77,53]).

Section 2 introduces preliminaries. In Sect. 3, we define *cPN*, present initial evidence for their modeling power and convenience, and show decidability of soundness. The discovery framework for *cPN* in Sect. 4 proves that by applying existing CPD techniques on *collaboration event logs* in the framework, we discover Petri nets that are *weakly bisimilar* to *cPN*. To underline favourable properties of *cPN*, we present empirical results that compare three publicly available CPD techniques on an existing dataset in Sect. 5. In Sect. 6, related work is discussed. Lastly, we conclude and give an outlook in Sect. 7.

2 Preliminaries

Let X be a set. $\mathcal{P}(X) = \{X' \mid X' \subseteq X\}$ denotes the *powerset* of X, and $\mathcal{P}^+(X) = \mathcal{P}(X) \backslash \emptyset$ (with $\emptyset$ the empty set) denotes the *set of all non-empty subsets* of X. Let Y be another set. A *relation* $R \subseteq X \times Y$ with $X \times Y = \{(x, y) \mid x \in X \wedge y \in Y\}$ the *cartesian product* of X and Y is a subset of all ordered pairs in the cartesian product. $\text{DOM}(R) = \{x \in X \mid \exists_{y \in Y} (x, y) \in R\}$ defines the *domain* of relation R and $\text{RNG}(R) = \{y \in Y \mid \exists_{x \in X} (x, y) \in R\}$ defines the *range* of relation R. Given set X', $R_{|X'} = \{(x, y) \in R \mid x \in X'\}$ defines the *restriction* of relation R's domain to X'. The *n-ary cartesian product* over sets $X_1, \ldots, X_n$ is defined by $X_1 \times \ldots \times X_n = \{(x_1, \ldots, x_n) \mid \forall_{i \in \{1,\ldots,n\}} x_i \in X_i\}$. We write $(x_i)_{i \in \{1,\ldots,n\}} \in X_1 \times \ldots \times X_n$ for an *n-tuple*, which is an element of the n-ary cartesian product over $X_1, \ldots, X_n$. If $X_1 = X, \ldots, X_n = X$, we also write X^n for $X_1 \times \ldots \times X_n$. Function application is naturally extended to sets, i.e., $f(X) = \{f(x) \mid x \in X\}$.

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 and $\emptyset$ also denotes the *empty multiset*, i.e., we overload notation. The *cardinality* of a multiset is defined by $|m| = \sum_{x \in X} m(x)$. 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. A set $X' \subseteq X$ is also the multiset $m(x') = 1$, $x' \in X'$. We also write $m = [x_1^{m(x_1)}, \ldots, x_n^{m(x_n)}]$ for $\text{supp(m)} = \{x_1, \ldots, x_n\}$. Given two multisets $m_1, m_2 \in \mathcal{B}(X)$, we lift the *subset relation*, *addition* and *subtraction* operation to multisets: (1) $m_1 \subseteq m_2$ iff for all $x \in X$ it holds that $m_1(x) \leq m_2(x)$; (ii) $(m_1 + m_2)(x) = m_1(x) + m_2(x)$ for each $x \in X$; and (iii) if $m_1 \subseteq m_2$, then $(m_2 - m_1)(x) = m_2(s) - m_1(x)$ for each $x \in X$.

A *trace* over X of *length* $n \in \mathbb{N}$ is a function $\sigma : \{1, \ldots, n\} \to X$. $|\sigma|$ denotes the length of trace σ. 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 $\exists_{i \in \{1,\ldots,|\sigma|\}} x = \sigma(i)$. The *support of trace* σ is $\text{supp}(\sigma) = \{x \in X \mid \exists_{1 \leq i \leq |\sigma|} \sigma(i) = x\}$. We define the *concatenation* $\sigma = \nu \cdot \gamma$ of two traces $\nu, \gamma \in X^*$ by $\sigma : \{1, \ldots, |\nu| + |\gamma|\} \to X$, $\sigma(i) = \nu(i)$ if $1 \leq i \leq |\nu|$ and $\sigma(i) = \gamma(i)$ if $|\nu| + 1 \leq i \leq |\nu| + |\gamma|$. 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.

Let Λ be a finite set of *activity labels*. A *labelled, directed process graph* is a 3-tuple (S, Λ, E) where S is the set of states, and $E \subseteq S \times \Lambda_\tau \times S$ the set of labelled edges, where $\Lambda_\tau = \Lambda \cup \tau$ for $\tau \notin \Lambda$ the *silent activity* [15,73]. For $(s, \alpha, s') \in E$, we also write $s \xrightarrow{\alpha} s'$. We write $s_1 \xrightarrow{\sigma} s_{n+1}$ iff there exists $\sigma \in \Lambda_\tau^*$ with $|\sigma| = n$ and $s_1, \ldots, s_{n+1} \in S$ such that $s_1 \xrightarrow{\sigma(1)} s_2 \ldots s_n \xrightarrow{\sigma(n)} s_{n+1}$. We define the *weak transition relation* $\stackrel{\alpha}{\Longrightarrow} \subseteq S \times \Lambda_\tau \times S$ with $\alpha \in \Lambda_\tau$ by (i) $s \, (\xrightarrow{\tau})^* \, s_1 \xrightarrow{\alpha} s_2 \, (\xrightarrow{\tau})^* \, s'$, if $\alpha \neq \tau$, and (ii) $s \, (\xrightarrow{\tau})^* \, s'$, if $\alpha = \tau$, where $(\xrightarrow{\tau})^*$ is the reflexive, transitive closure of $\xrightarrow{\tau}$. We lift the notation of the weak

transition relation to traces and write $s_1 \stackrel{\sigma}{\Longrightarrow} s_{n+1}$, iff there exists $\sigma \in \Lambda^*$ with $|\sigma| = n$ and $s_1, \ldots, s_{n+1} \in S$ such that $s_1 \stackrel{\sigma(1)}{\Longrightarrow} s_2 \ldots s_n \stackrel{\sigma(n)}{\Longrightarrow} s_{n+1}$.

A *labelled transition system* (LTS) over Λ is a labelled, directed process graph with an initial state: $\Gamma_\Lambda = (S, \Lambda, s_0, \longrightarrow)$, where s_0 is the *initial state*. We define the equivalence relation (rooted) *weak bisimulation equivalence* on the set of all LTS over Λ. Let $\Gamma_1 = (S_1, \Lambda, s_{0,1}, \longrightarrow_1)$ and $\Gamma_2 = (S_2, A, s_{0,2}, \longrightarrow_2)$ be two LTSs. A relation $R \subseteq S_1 \times S_2$ is a (rooted) *weak bisimulation equivalence*, denoted by $\Gamma_1 \approx_R \Gamma_2$, iff (i) $(s_{0,1}, s_{0,2}) \in R$, i.e., the initial states are related; and for every $(p, q) \in R$ and $\alpha \in \Lambda_\tau$ it holds that (ii) if $p \stackrel{\alpha}{\longrightarrow}_1 p'$, then $\alpha = \tau$ and $(p', q) \in R$, or there exists $q' \in S_2$ such that $q \stackrel{\alpha}{\Longrightarrow}_2 q'$ and $(p', q') \in R$; and (iii) if $q \stackrel{\alpha}{\longrightarrow}_2 q'$, then $\alpha = \tau$ and $(p, q') \in R$, or there exists $p' \in S_1$ such that $p \stackrel{\alpha}{\Longrightarrow}_1 p'$ and $(p', q') \in R$. If $\Gamma_1 \approx_R \Gamma_2$, we say Γ_1 is weakly bisimilar to Γ_2.

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 tuple (N, m) is called a *marked Petri net*. Given labelled Petri net N, $\mathcal{N}(N) = \{(N, m) \mid m \in \mathcal{B}(P)\}$ denotes the set of all marked Petri nets of N. 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, which remove a token from each of it's input places, adds a token to each of it's output places, and executes an activity $\alpha \in \Lambda_\tau$ represented by $l(t)$. We define this behavior as a relation called the *firing rule* $\cdot[\cdot\rangle\cdot \subseteq \mathcal{N}(N) \times \Lambda_\tau \times \mathcal{N}(N)$ by $(N, m)[l(t)\rangle(N, m')$ iff $(N, m)[t\rangle$ and $m' + \bullet t = m + t\bullet$.

The semantics of a marked Petri net (N, m_0) with $N = (P, T, F, l, \Lambda_\tau)$ is defined by the LTS $\Gamma_{N,m_0} = (\mathcal{B}(P), \Lambda_\tau, m_0, \longrightarrow)$ with $m \stackrel{\alpha}{\longrightarrow} m'$ iff $\alpha = l(t)$ and $(N, m)[l(t)\rangle(N, m')$. (N, m_0) is weakly bisimilar to marked Petri net (N', m_0') iff their semantics are. A trace $\sigma \in \Lambda_\tau^*$ is a *firing trace* of (N, m_0) iff $m_0 \stackrel{\sigma}{\longrightarrow} m'$. We omit the labelled Petri net N, if the context is clear. We define the set of *reachable markings* by $\mathcal{R}(N, m_0) = \left\{m' \mid \exists_{\sigma \in \Lambda_\tau^*}\, m_0 \stackrel{\sigma}{\longrightarrow} m'\right\}$. Given marking $m' \in \mathcal{B}(P)$, we denote the set of all finite firing traces that reach m' in (N, m_0) by $\mathcal{L}_\tau(N, m_0, m') = \left\{\sigma \in \Lambda_\tau^* \mid m_0 \stackrel{\sigma}{\longrightarrow} m'\right\}$. The *trace language* only contains observable activities $a \in \Lambda$ and is defined by $\mathcal{L}(N, m_0) = \left\{\sigma \in \Lambda^* \mid m_0 \stackrel{\sigma}{\Longrightarrow} m'\right\}$. A marked Petri net (N, m_0) is *bounded* iff its set of reachable markings $\mathcal{R}(N, m_0)$ is finite. (N, m_0) is *live* iff for every transition $t \in T$ and reachable marking $m \in \mathcal{R}(N, m_0)$, there exists a reachable marking $m' \in \mathcal{R}(N, m)$ such that $(N, m')[t\rangle$. A transition $t \in T$ is *dead* iff there is no reachable marking $m \in \mathcal{R}(N, m_0)$ such that $(N, m)[t\rangle$. N is a *workflow net* (WF-net) iff (i) there exists a single source place $i \in P$ such that its preset is empty: $\bullet i = \emptyset$; (ii) there exists a single sink place $o \in P$ such that its postset is empty: $o\bullet = \emptyset$; and (iii) every node $x \in P \cup T$ is on a directed path from i to o. N is *sound* iff (i) it is *weakly terminating* (also called option to complete), i.e., for all reachable markings

$m \in \mathcal{R}(N, [i])$, $[o] \in \mathcal{R}(N, m)$; (ii) it is *proper completing*, i.e., for all reachable markings $m \in \mathcal{R}(N, [i])$, if $[o] \subseteq m$, then $m = [o]$; and (iii) it has no dead transitions.

3 Collaboration Petri Nets

In this section, we introduce collaboration Petri nets (cPN) as a standardized Petri net class to model collaboration processes. cPN are designed to reuse concepts, patterns, and analysis techniques from existing work on collaboration processes and process orchestrations such that we can apply the vast existing body of knowledge and tools without modifications. As we will show, cPN are WF-nets, i.e., we can recursively model and discover collaboration processes at various granularity levels. Consequently, our model of collaboration processes is equivalent to process orchestrations in its modeling and decision power, a result that is not obvious from the multitude of Petri net extensions proposed in existing work. With respect to modeling convenience, cPN explicitly represent all four of the known collaboration types [17] similar to *RM_WF_nets* [77,76,53], in a declarative, concise, standardized, modular, and correct definition.

3.1 Building Blocks of Collaboration Processes

To model collaboration processes, we first identify relevant building blocks that we formalize step-by-step in this section to finally construct cPN in the next section. To begin with, collaboration processes are studied in various research areas [17]. Research areas with a modeling focus are *process choreographies* [28,29,37,55,38], *service compositions* [66,9,7], *multi-agent systems* [70,71], *interorganizational workflows* [5,11], *virtual enterprises* [22,44,43], and *distributed business processes* [19,68], while *object-centric process mining* [8,15] and *collaboration mining* [40,77,69,75,24,50,59,53,72] are research areas with a discovery focus. Despite the variety of areas and approaches, collaboration processes are conceptualized as a set of *collaboration concepts* whose process orchestrations collaborate. Below, we formalize collaboration concepts $\mathcal{C}$ and the collaboration concepts involved in a single collaboration process.

Definition 1 (Collaboration Concept, Concept Collection). *Let $\mathcal{C}$ be a set of collaboration concepts. We define a (collaboration) concept collection cc of length $n \in \mathbb{N}$ to be an injective trace over $\mathcal{C}$: $cc : \{1, \ldots, n\} \to \mathcal{C}$.*

By means of the concept collection cc, we group the subset of collaboration concepts that are involved in a collaboration process, e.g., $cc(1)$ = "emergency", $cc(2)$ = "X-ray" with $n = 4$ in Fig. 1. Because each collaboration concept's dynamic behavior is a process orchestration, every collaboration concept $c \in \text{RNG}(cc)$ is assigned a WF-net that models the collaboration concept's process orchestration. We formalize this notion as a *workflow collection*:

Definition 2 (Workflow Collection). *Let cc be concept collection. A workflow collection is a $|cc|$-tuple $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ of WF-nets $N_c = (P_c, T_c, F_c, l_c, \Lambda_\tau)$ such that $\forall_{k,l \in \{1,\ldots,n\}}$ if $k \neq l$, then $\left(P_{cc(k)} \cup T_{cc(k)}\right) \cap \left(P_{cc(l)} \cup T_{cc(l)}\right) = \emptyset$. We define:*

- $T^u = \bigcup_{c \in \mathrm{RNG}(cc)} T_c$, $P^u = \bigcup_{c \in \mathrm{RNG}(cc)} P_c$, $F^u = \bigcup_{c \in \mathrm{RNG}(cc)} F_c$, *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* $t \in T_c$.

Existing work on collaboration processes shares the conceptualization (modulo naming) up to this point, i.e., collaboration processes are modelled as a set of WF-nets to the best of our knowledge. Since the WF-nets in a collaboration process collaborate, they must be composed to represent the overall collaboration process. Each existing approach models the composition in a certain way, but they all are formalized according to the way WF-nets collaborate, i.e., the type of composition is determined by the *collaboration type* studied.

There are four collaboration types: Handover-of-work, message exchange, resource sharing, and activity execution [76,17]. The first three types are asynchronous, and the fourth is synchronuous as depicted with their respective, typical Petri net pattern to model the collaboration type in Fig. 2. For example, the discovery techniques in [76,54,53,72] represent handing the treatment of patients over in healthcare collaboration processes using the depicted handover-of-work pattern. The pattern includes an asynchronuous collaboration place p_{ac} (denoted by a grey filling) that must be marked before the first transition in $N_{cc(1)}$ can fire. The handover-of-work type is a special case of the message exchange type and determined by condition (HO) given some collaboration concept $c \in \mathrm{RNG}(cc)$, a transition $t \in i_c\bullet$ in the postset of source place i_c in WF-net N_c has the asynchronous collaboration place p_{ac} in its preset $p_{ac} \in \bullet t$ [76]. Although the

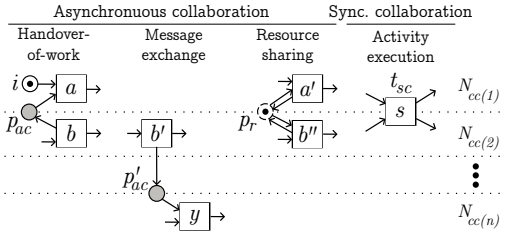

Fig. 2. Petri net patterns to model collaboration between process orchestrations.

resource sharing pattern with place p_r (denoted with dashed outline in Fig. 1 and Fig. 2) does not affect the behavior of the models in [77,76,53] due to the lack of a time notion, it can be seen as a precursor of mining enhanced models that introduce a time notion such as the *healthcare Petri net* with time delays for transition firings in [54]. We follow the same path and keep a time notion out of

our formalizations bearing in mind that it can be easily added later. Lastly, the activity execution pattern includes a transition t_{sc} whose firing is synchronized across multiple WF-nets, as, for example, employed in [5,9,35,15]. Since our aim is to standardize and integrate, we cover all existing collaboration types in a *collaboration pattern* as follows.

Definition 3 (Collaboration Pattern). *Let $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ 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., $P_{AC} \cap (P^u \cup T^u) = \emptyset$,*
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 channel $p_{ac} \notin P_{RS}$ denotes that transitions $t \in T_s$ send a message and transitions $t' \in T_r$ receive a message 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.*

A collaboration pattern CP_{WC} specifies how the WF-nets in the workflow collection collaborate in terms of the four collaboration types. All three asynchronuous collaboration types are represented by P_{AC} (cf. 1 in Def. 3) and AC (4). The resource sharing type is differentiated by P_{RS} (2), the resource allocation ra, and the self-loop requirement in (5) from the other two asynchronous types. Message exchange and handover-of-work are not explicitly distinguished in the collaboration pattern, because their only difference is captured by condition (HO). Although synchronized transitions $t \in T^u$ with $t \in T_{sc}, (t_{sc}, T_{sc}) \in ET$ are pre-determined by equal labels (6) following CPD techniques with synchronous collaboration [60,77,65,8,59,53,15], we still formalize them in the collaboration pattern for the sake of completeness.

A collaboration pattern CP is powerful and convenient in representing collaborations (see Sect. 4). For example, CP can represent many *web service interaction patterns* [16] as message exchanges along the dimensions of # of services involved in an exchange (T_s or T_r may contain transitions from an arbitrary number of collaboration concepts), the # of exchanges (transitions in T_s or T_r may be in a loop in their respective WF-net N_c), and the difference between *round-trip* interactions (two channels p_{ac} with alternating sender and receiving collaboration concepts) and *routed* interactions (a transition is allowed to be both a receiver from one and sender to another collaboration concept).

In the following section, we will compose the WF-nets in the workflow collection by means of the collaboration pattern into a collaboration Petri net.

3.2 Collaboration Petri Nets

The definition of the collaboration Petri net given a workflow collection and collaboration pattern takes the same definition approach as the *interorganizational workflow* in [5] with extensions and modifications to cover all four collaboration types. The modifications and extensions are defined such that the structure of the resulting Petri net remains a WF-net, which we will show in the next section.

Definition 4 (Collaboration Petri Net (cPN)). *Let $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration pattern with $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ a workflow collection. A Collaboration Petri Net is a marked Petri net $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ defined as*[1]*:*

1. $P = P^u \cup P_{AC} \cup \{i, o\}$,
2. $T = r\,(T^u) \cup \{t_i, t_o\}$, *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. $F' = F^u \cup \{(t, p) \in T^u \times P_{AC} \mid (p, x, y) \in AC \wedge t \in x\} \cup \{(p, t) \in P_{AC} \times T^u \mid (p, x, y) \in AC \wedge t \in y\} \cup \{(i, t_i), (t_o, o)\} \cup \{(t_i, i_c) \mid c \in \mathrm{RNG}(cc)\} \cup \{(o_c, t_o) \mid c \in \mathrm{RNG}(cc)\}$,
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. 1 is indeed a cPN by definition. Aside from these properties of cPN that mostly determine modeling convenience, our approach to separate the WF-nets in the workflow collection, from their collaborations in the collaboration pattern, and their resulting collaboration process model in the cPN resembles, yet clarifies the compositional approach commonly taken for modeling and discovery of collaboration processes.

In the last section, we have informally introduced the four collaboration types and depicted their Petri net patterns in Fig. 2. Below, we formalize them:

Definition 5 (Collaboration Types). *Let $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration pattern with $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ a workflow collection. We define $\Upsilon = \{\upsilon_s, \upsilon_m, \upsilon_h, \upsilon_r\}$ the set of collaboration types. Then,*

- *Activity execution, denoted by υ_s, is contained in CP iff $ET \neq \emptyset$,*
- *Message exchange, denoted by υ_m, is contained in CP iff $P_{AC} \setminus P_{RS} \neq \emptyset \wedge \exists_{(p, T_s, T_r) \in AC}\ p \in P_{AC}$,*
- *Handover-of-work, denoted by υ_h, is contained in CP iff it contains υ_m such that the handover condition (HO) holds, and*
- *Resource sharing, denoted by υ_r, is contained in CP iff $P_{RS} \neq \emptyset$.*

[1] We choose to abbreviate collaboration Petri nets with cPN to mitigate confusion with the abbreviation cPN of colored Petri nets [49].

We say that collaboration Petri net $cPN = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ *contains collaboration types* $\Upsilon' \subseteq \Upsilon$ *iff every collaboration type* $\upsilon \in \Upsilon'$ *is contained in collaboration pattern* CP*.*

Hence, a collaboration pattern can represent all four collaboration types and in the collaboration Petri net are exactly the Petri net patterns as depicted in Fig. 2, e.g., $t_7, p_{ac,1}, t_{16}$ is the Petri net pattern for handover-of-work in Fig. 1.

Multiplicities represented by arc weights and other higher-level net concepts such as *token colors* [49,64] or *variable arcs* [8] are missing. Avoiding higher-level net concepts strikes a balance between the four requirements for a standard (cf. Sect. 1). On the one hand, modeling power is slightly reduced such that collaboration processes in which multiple process orchestration instances of a collaboration concept are required for a single execution of the collaboration processes (cf. [76,8]) cannot be represented. Hence, token colors are not needed to distinguish the process instances running in the same process orchestration, as there is only one. The resulting *representational bias* [3] is chosen by the majority of modeling and discovery techniques [5,28,40,9,77,54,75,51,69,50,59,53,72], which we take as evidence that our standard's representational bias is acceptable for most collaboration processes and domains (cf. Tab. 1). On the other hand, more relevant problems are decidable and likely fall into a more desirable *complexity* class. Moreover, a cPN is likely to be more understandable due to its restriction on basic net concepts, i.e., modeling convenience for practitioners is better.

Interestingly, *typed Jackson nets* [15] employ token colors in the form of identifiers to represent the collaboration process in a continuously running state, but still require a one-to-one correspondence between types in synchronous collaborations within their discovery framework. A cPN models the collaboration process in a static state, i.e., for a single execution similar to WF-nets for process orchestrations. The next section shows that a cPN is a WF-net, from which we conclude decidability of soundness.

3.3 Automated Verification of Collaboration Petri Nets

We start by showing that a cPN is a WF-net:

Lemma 1. *Let* $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ *be a collaboration Petri net.* cPN *is a WF-net.*

Proof. It follows from Def. 4 (1), (3), (4), and (5) that the only source place i and sink place o are connected to the source places i_c and o_c of the workflow nets N_c for $c \in \mathrm{RNG}(cc)$ via t_i and t_o respectively. Hence, any node in workflow net N_c lies on a directed path from i to o. Moreover, asynchronous places $p_{ac} \in P_{AC}$ with $(p_{ac}, T_s, T_r) \in AC$ lie on a directed path from transitions $t \in T_s$ to transitions $t' \in T_r$ as follows from Def. 4 (4) and (5). Fused transitions $t \in T_{sc}$ for some $(t_{sc}, T_{sc}) \in ET$ lie on a directed path within all their fused collaboration concept's WF-nets N_c as follows from Def. 4 (2), (4), and (5). ■

As mentioned in Sect. 3.1, resource places p_r leave behavior unaffected without a time notion in our model, but still render the classical soundness definition inapplicable. However, soundness is meaningful in our untimed setting, which is expressed in a sightly changed soundness definition for cPN.

Definition 6 (Sound cPN). *Let* $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ *be a collaboration Petri net. We say* cPN *is sound iff the WF-net resulting from removing all shared resource places* $p_r \in P_{RS}$ *and their arcs through the "elimination of self-loop places" reduction rule [56] is sound.*

We are able to verify soundness of collaboration Petri nets.

Theorem 1 (Decidability of Soundness for cPN). *Soundness for the collaboration Petri net* $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ *is decidable.*

Proof. Observe that the reduction rule "elimination of self-loop places" preserves liveness, safety and boundedness [56] and that reducing the cPN to a collaboration Petri net without any shared resource places does not break the WF-net structure, as the reduction is similar to constructing cPN with collaboration pattern CP that does not contain any shared resources, i.e., $P_{RS} = \emptyset$. Since soundness is decidable for WF-nets through reducing it to deciding *boundedness* and *liveness* in the *short-circuited* WF-net in which the sink place is connected to the source place via a transition labelled with τ [2], it follows from Lemma 1 that soundness is decidable for collaboration Petri nets.

■

To sum up, cPN have sufficient modeling power to represent all collaboration types and soundness is decidable. We can reuse existing results and techniques developed for modeling and analysing process orchestrations also for collaboration processes by avoiding higher-level net concepts and other structural extensions. On top, we can recursively stick a cPN into another workflow collection at a higher granularity level such that we can seamlessly model collaboration at any granularity level. Next, we move from modeling to discovering cPN.

4 Discovering Collaboration Petri Nets

Figure 3 presents the overall framework to discover collaboration Petri nets cPN (all related concepts are defined in the following sections): Given *collaboration event log* cL converted to cL' in the required input format of CPD technique *discover*, converted log cL' is projected to each collaboration concept's event log $L_{cc(1)}, \ldots, L_{cc(n)}$. Then, process discovery technique *disc* discovers WF-nets $N_{cc(1)}, \ldots, N_{cc(n)}$, while some custom mining functionality *cdisc* mines asynchronous collaborations as Petri net patterns resulting in N_{ac} (synchronous collaborations are always determined by equal labels). All results are composed with some custom model composition $comp(N_{cc(1)}, \ldots, N_{cc(n)}, N_{ac}) = N$. If *discover* has property $Q_{discover}$, we can construct WF-nets $N_{cc_{ac}(1)}, \ldots, N_{cc_{ac}(n)}$ to simulate N_{ac} and N is weakly bisimilar to collaboration Petri net cPN. Because

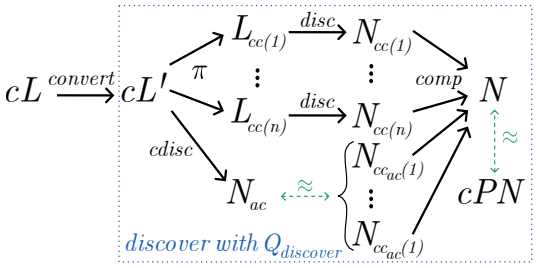

Fig. 3. Our framework to discover collaboration Petri net cPN from collaboration event log cL. If CPD technique *discover* has property $Q_{discover}$, then $N \approx cPN$.

many existing CPD techniques (cf. Sect. 5) have $Q_{discover}$, collaboration Petri nets achieve comparability between CPD techniques despite the heterogeneity of collaboration concept types, collaboration types, and discovered models.

4.1 Generating Collaboration Event Logs

In short, a *collaboration event log* cL is generated by a collaboration Petri net $cPN = (N, m_0)$ that models the real-world collaboration process[2]. If we only record the trace of (observable) activity labels $\sigma \in \Lambda^*$ in a firing trace $(N, m_0) \overset{\sigma}{\Longrightarrow} (N, m)$ for some $m \in \mathcal{R}(N, m_0)$ to generate a collaboration event log, we do not have any information on collaborations, i.e., synchronous activities, message exchanges and resource requirements are unknown. To additionally record information on collaborations during execution of a cPN, we propose the collaboration labelling l_{cl} that replaces l in a cPN.

Definition 7 (Collaboration Labelling). *Let $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration pattern with $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ a workflow collection. Collaboration labelling is a function $l_{cl} : T^u \cup \{t_i, t_o\} \to \Lambda_{cl}$ with collaboration labels $\Lambda_{cl} = \Lambda_\tau \times \mathcal{P}(\mathrm{RNG}(cc)) \times \mathcal{P}(P_{AC})^3$ for $t_i, t_o \notin T^u$:*

$$l_{cl}(x) = \begin{cases} (l^u(t), sync(T_{sc}), send(x), rec(x), res(x)) & \text{if } (x, T_{sc}) \in ET \wedge t \in T_{sc} \\ (l^u(x), \quad \{c\}, \qquad send(x), rec(x), res(x)) & \text{if } (x, T_{sc}) \notin ET \wedge \\ & \exists_{c \in \mathrm{RNG}(cc)}\ x \in T_c, \\ \tau & \text{otherwise.} \end{cases}$$

where

$$\begin{aligned} sync(T_{sc}) &= \{c \in \mathrm{RNG}(cc) \mid t \in T_{sc} \wedge t \in T_c\} \\ send(x) &= \{p_{ac} \mid \exists_{(p_{ac}, T_s, T_r) \in AC}\ x \in T_s \wedge p_{ac} \notin P_{RS}\}, \\ rec(x) &= \{p_{ac} \mid \exists_{(p_{ac}, T_s, T_r) \in AC}\ x \in T_r \wedge p_{ac} \notin P_{RS}\}, \text{and} \\ res(x) &= \{p_r \mid \exists_{(p_r, T_1, T_2) \in AC}\ x \in T_1 \wedge p_r \in P_{RS}\}. \end{aligned}$$

[2] A cPN used to generated cL is a reference model with which we can compare model quality metrics computed on cPN and a discovered cPN', i.e., our framework builds the foundation to study the *maturity* of CPD techniques [74]

We call $(activity, concepts, send, receive, resource) \in \Lambda_{cl}$ *a collaboration label.*

With the collaboration labelling l_{cl}, we can generate collaboration event logs cL given a collaboration Petri net cPN as follows:

Definition 8 (Collaboration Event Log). *Given a set of activity labels* Λ*, a set of traces* $L \subseteq \Lambda^*$ *is called an event log. An event log is generated by a marked Petri net* (N, m_0)*, if* $m_0 \xRightarrow{\sigma} m$ *with* $m \in \mathcal{B}(P)$ *for all* $\sigma \in L$*, i.e., an event log is a subset of the trace language* $L \subseteq \mathcal{L}(N, m)$*. For a collaboration Petri net* $cPN = ((P, T, F, l_{cl}, \Lambda_{cl}), m_0)$ *with collaboration labelling, we call event log* $cL \subseteq \Lambda_{cl}^*$ *generated by* cPN*, i.e.,* $cL \subseteq \mathcal{L}(cPN)$*, a collaboration event log.*

Note that the information recorded in cL is on the highest logging level according to logging levels specified in [67,32,31,41] for discovery of web service compositions, i.e., collaboration processes. The highest logging level is sufficient to discover the complete, executable collaboration process by some CPD technique *discover* [50], if no multiplicities between collaboration concept's WF-net instances are present (cf. Sect. 3.2).

We can *convert* cL into an *eXtensible Event Stream* (XES) [1] event log cL' (cf. Fig. 3) in which a collaboration label's *activity* corresponds to the `concept:name`, it's *concepts* correspond to the `org:resource`, it's *send* corresponds to the `message:sent`, it's *receive* corresponds to the `message:rec`, and it's *resource* to the `resource` event attributes as specified in [53]. Also, we give an intuition on how we can *convert* cL into an object-centric event log (OCEL) cL' by defining *object types* for each collaboration concept $c \in \text{RNG}(cc)$ and each asynchronous type $p_{ac} \in P_{AC}$. Each event's *object map* denoted by $omap(e)$ in the OCEL cL' that maps object types to *object identifiers* such that the respective objects are associated with the event has to be set. Each object type can be taken as a case notion [8]. As $\sigma_j(i) \in \Lambda_{cl}$ for $\sigma_j \in cL, i \in \{1, \ldots, |\sigma_j|\}$ corresponds to an event in cL', set $omap(e)(c) = j$ if $c \in concepts_{i,j}$, $omap(e)(p_{ac}) = \langle p_{ac}, j, \#_j(p_{ac}, mode)\rangle$[3] if $p_{AC} \in send_{i,j} \cup receive_{i,j}$ and $p_{AC} \in P_{AC} \setminus P_{RS}$, and $omap(e)(p_{AC}) = j$ if $p_{AC} \in resource_{i,j}$ and $p_{AC} \in P_{RS}$ in ascending order of i. Overall, it is important to observe that the conversions are bijections without information loss such that we can convert cL' back into cL.

As CPD techniques *discover* typically employ a divide-and-conquer approach to discover collaboration process models M from cL as depicted in Fig. 3, the first step of *discover* is a *log projection* as follows.

Definition 9 (Log Projection). *Let* $cL \subseteq \mathcal{L}(cPN)$ *for collaboration Petri net* cPN *be a collaboration event log. Log projection is defined by* $\pi_c(cL) = \{\sigma_c \mid \sigma \in cL \wedge \sigma_c = \sigma_{|\Lambda_\tau \times \{X \subseteq \mathbb{P}(\text{RNG}(cc)) | c \in X\} \times \mathbb{P}(P_{AC})^3}\}$ *for* $c \in \text{RNG}(cc)$*.*

In the next section, we show that the asynchronous collaboration types as represented by P_{AC} and AC are syntactic sugar for the synchronous collaboration type represented by ET in a collaboration pattern CP.

[3] $\#_j$ is a counter function counting how often a message of type p_{AC} is sent/received as passed by argument *mode* in σ_j such that a first-in first-out message queue is assumed.

4.2 Minimal Collaboration Types

Taking the modeling power point of view, collaboration types can be simulated by another, i.e., we can reduce the number of collaboration types to a *minimal* set while maintaining weak bisimilarity between the respective collaboration Petri nets. We show that synchronous collaboration is minimal. To prove our statement, a relation on the set of collaboration types Υ is required. We aim for a relation that at least partially orders the powerset of collaboration types $\mathcal{P}(\Upsilon)$ such that Υ is the maximal element and, ideally, a unique minimal element exists. A minimal element Υ' means that all other collaboration types $\upsilon \in \Upsilon \setminus \Upsilon'$ are syntactic sugar and are included in collaboration Petri nets to improve the modeling convenience.

Definition 10 (Simulating Collaboration Types, Minimality). *A non-empty set of collaboration types $\Upsilon_1 \subseteq \Upsilon$ simulates collaboration types $\Upsilon_2 \subseteq \Upsilon$ with $\Upsilon_1 \subseteq \Upsilon_2$, denoted by $\Upsilon_1 \preccurlyeq \Upsilon_2$ iff for any collaboration Petri net cPN that contains collaboration types $\upsilon \in \Upsilon_2$ and is defined on workflow collection $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$, there exists a collaboration Petri net cPN' that only contains types $\upsilon' \in \Upsilon_1$ and is defined on $WC' = (N_c)_{c \in \mathrm{RNG}(cc')}$ with $cc' = cc \cdot cc''$ such that $cPN \approx cPN'$. A subset $\Upsilon_m \subseteq \Upsilon$ of collaboration types is minimal iff collaboration types Υ_m simulate Υ and there exists no other set of collaboration types Υ'_m with $|\Upsilon'_m| < |\Upsilon_m|$ such that collaboration types Υ'_m simulate Υ.*

In the construction of a cPN' that simulates cPN, it is important to prohibit changes to the workflow collection $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ except than extending to WC', as other changes to WC would mean that we allow simulating collaboration types by a collaboration concept's internal WF-net and not by some other collaboration type. Note that $\preccurlyeq \subseteq \Upsilon \times \Upsilon$ defines a partial order on collaboration types. In the following, we show that the synchronous collaboration type is a minimal element of $\preccurlyeq$, under two conditions: the empty trace is not in the trace language and no *τ-labelled loop* for asynchronous collaboration exists. A *τ-labelled loop* in $cPN = (N, m_0)$ exists for asynchronous collaboration $(p_{ac}, T_s, T_r) \in AC$ iff $m_1 \xrightarrow{l(t)} m_2$, $m_1 \in \mathcal{R}(N, m_0)$ and $m_2 \xRightarrow{\tau} m_3$ such that $(N, m_3)[t'\rangle$ for some $t, t' \in T_s$. The following theorem proves our statement with an efficient construction in the sense that the constructed, synchronous collaboration only cPN_{os} that is weakly bisimilar to a given cPN does not copy complete WF-nets of collaboration concepts.

Theorem 2 (Synchronous activity execution is minimal). *$\Upsilon' = \{\upsilon_s\}$ is minimal, if no asynchronous collaboration $(p_{ac}, T_s, T_r) \in AC$ with τ-*labelled loops *in cPN exists and $\epsilon \notin \mathcal{L}(cPN)$.*

Proof. Let $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus^{CP}_{c \in \mathrm{RNG}(cc)} N_c$ be a collaboration Petri net that contains collaboration types $\upsilon \in \Upsilon$ with $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ a collaboration pattern and $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ a workflow collection. Observe that labelling of transitions with observable activities $\alpha \neq \tau$, $\alpha \in \Lambda_\tau$ is a bijection in any collaboration Petri net (cf. Def. 3), so we can use activity

α and its transition interchangeably. In the following we construct collaboration Petri net cPN_{os} of only synchronous collaboration by lifting asynchronous collaboration to a collaboration concept, i.e., every message channel and resource type becomes a collaboration concept. Let $M_f(cPN) = \{m \in \mathcal{B}(P) \mid m(o) \geq 1\} \cap \mathcal{R}(cPN)$ be the set of markings that mark the global sink place for cPN. We preserve reaching the global sink place, i.e., $M_f(cPN) \neq \emptyset$ iff $M_f(cPN_{os}) \neq \emptyset$. Hence, we distinguish three cases of asynchronous collaboration. Given $(p_{ac}, T_s, T_r) \in AC$, either (i) p_{ac} is *dead*, i.e., p_{ac} is *dead* iff all $t \in T_s$ are dead or for every firing trace $\sigma \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_\tau(N, m_0, m)$ that marks the global sink place it holds that $\forall_{\alpha \in l(T_s)}\ \alpha \notin \sigma$; (ii) p_{ac} is *optional*, i.e., p_{ac} is *optional* iff there exists at least two firing traces $\sigma, \sigma' \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_\tau(N, m_0, m)$ that mark the global sink place such that $\forall_{\alpha \in l(T_s)}\ \alpha \notin \sigma$ (no sending activity occurs in σ) and $\exists_{\alpha \in l(T_s)}\ \alpha \in \sigma'$ (a sending activity occurs in σ'); and (iii) p_{ac} is *compulsory*, i.e., p_{ac} is *compulsory* iff for every firing trace that marks the global sink place $\sigma \in \bigcup_{m \in M_f(cPN)} \mathcal{L}_\tau(N, m_0, m)$ it holds that $\exists_{\alpha \in l(T_s)}\ \alpha \in \sigma$ (at least one sending activity always occurs). Note that we can distinguish these three cases by analysing the *coverability graph* [56,62]. However, we do not have to determine for each asynchronous collaboration what case applies, as we only have to compute certain markings with respective sets of activities in the coverability graph that we will define in the following. We can ignore case (i) to construct a weakly bisimilar cPN_{os}, as the sending activities of p_{ac} can never occur in any firing trace. Due to (ii), we have to ensure that the WF-net $N_{p_{ac}}$ that simulates optional asynchronous collaboration p_{ac} with synchronous collaboration is able to skip all transitions with sending activities, as otherwise sink place $o_{p_{ac}}$ cannot be marked such that $[o_{os}]$ becomes unreachable. Skipping has to occur synchronously with activities "after" which sending cannot occur anymore. Also, repeating sending activities by some loop has to be allowed and synchronized with activities "after" which sending occurs again.

Determine by breadth-first search for each (p_{ac}, T_s, T_r) a unique set of markings $M_\times \subseteq \mathcal{R}(N, m_0)$ and set of markings $M_{\circlearrowleft} \subseteq \mathcal{R}(N, m_0)$ in the coverability graph. $M_\times$ is characterized by the following formula: for each $m_\times \in M_\times$ it holds that $m_\times$ is reachable by paths in the coverability graph in which no sending activities $\alpha \in l(T_s)$ have occurred and formula $\gamma(m_\times)$ holds. $\gamma(m)$ holds iff there exist two sets of activities $A_0, A_1 \subseteq \Lambda_\tau$, $\tau \notin A_1$ with $m \overset{\alpha_0}{\Longrightarrow} m_\neg$, $\alpha_0 \in A_0$ such that for all $m' \in \mathcal{R}(N, m_\neg)$ sending activities cannot occur $\neg(N, m')[t\rangle$ for all $t \in T_s$ and $m \overset{\alpha_1}{\Longrightarrow} m_1$, $\alpha_1 \in A_1$ such that there exists $m' \in \mathcal{R}(N, m_1)$ that enables sending activities $(N, m')[t\rangle$ for some $t \in T_s$. Thus, the activities A_0 and A_1 represent the choice between not executing optional asynchronous collaboration p_{ac} and executing asynchronous collaboration p_{ac}. It follows from $\epsilon \notin \mathcal{L}(cPN)$, that at least one $m_\times$ always exists such that $M_\times$ cannot be empty, if p_{ac} is optional. Let $A'_\times = \bigcup_{m_\times \in M_\times} A_{0,m_\times}$ be the union of activities $A_{0,m_\times}$ that are computed to satisfy $\gamma(m_\times)$. Define $T'_{p_{ac},\times} = \{t \in T \mid l(t) \in A'_\times \setminus \{\tau\}\} \cup \{t_{p_{ac},\tau} \mid \tau \in A'_\times\}$ for $t_{p_{ac},\tau} \notin T$ the set of transitions for asynchronous collaboration p_{ac} that, if executed, can only result in firing traces without any sending activity. Observe that $T_{p_{ac},\times} = \emptyset$ in case of compulsory asynchronous collaboration (iii).

$M_{\circlearrowleft}$ is characterized by the following formula: for each $m_{\circlearrowleft} \in M_{\circlearrowleft}$ it holds that $m_{\circlearrowleft}$ is reachable by paths in the coverability graph in which some sending activity $\alpha \in l(T_s)$ has occurred and (i) $\gamma(m_{\circlearrowleft})$ holds or (ii) there exists a *token generator* [34] trace $\sigma \in \Lambda_\tau^*$ with sending activity α in the trace $\alpha \in \sigma$. A token generator trace results in unbounded states of the coverability graph (ω-states) and is characterized by: There exists $m \in \mathcal{B}(P)$ such that $|m| > 0$ and $m_{\circlearrowleft} \xrightarrow{\sigma} m_{\circlearrowleft}+m$. If (ii) holds, the transition $t \in T_s$ labelled with the sending activity $l(t) = \alpha$ can fire infinitely often. Define $A_{1,m_{\circlearrowleft}} = \{\alpha_1\}$ with activity $\alpha_1 \in \sigma, \alpha \neq \alpha_1$ some activity from the token generator trace that always occurs in the trace, i.e., that is part of the cycle in the coverability graph that leads to the unbounded state. If a sending activity is repeated, the set $M_{\circlearrowleft}$ cannot be empty and $A_{1,m_{\circlearrowleft}}$ is always well-defined in case of (ii) for some sending activity, since cPN does not contain any *τ-labelled loops* for all $(p_{ac}, T_s, T_r) \in AC$. Note that the distinction between a choice of repeating sending activity again (i) or infinitely often repeating a sending activity (ii) is exhaustive, as duplicate labels, i.e., $t, t' \in T$ with $l(t) = l(t')$ cannot exist by definition. Let $A_{\circlearrowleft} = \bigcup_{m_{\circlearrowleft} \in M_{\circlearrowleft}} A_{1,m_{\circlearrowleft}}$ be the union of activities $A_{1,m_{\circlearrowleft}}$ that are computed to satisfy $\gamma(m_{\circlearrowleft})$. These activities signify that sending activities are executed again. Let $T_{p_{ac},\circlearrowleft} = \{t \in T \mid l(t) \in A_{\circlearrowleft}\}$ be the set of transitions for asynchronous collaboration p_{ac} that, if executed, result in firing traces with repeated sending activities. Observe that $T_{p_{ac},\circlearrowleft} = \emptyset$ in case of no loops involving sending activities.

Since $T'_{p_{ac},\times}$ may not yet include transitions that indicate that sending activities are not sent again in a loop, we add them as follows. Let $A_{0,m_{\circlearrowleft}}$ for $m_{\circlearrowleft} \in M_{\circlearrowleft}$ be the sets of activities that were computed to satisfy $\gamma(m_{\circlearrowleft})$, i.e., an activity $\alpha \in A_{0,m_{\circlearrowleft}}$ indicates that after some sending activity has occurred, it will not occur again. Next, $A_\times = A'_\times \cup \bigcup_{m_{\circlearrowleft} \in M_{\circlearrowleft}} A_{0,m_{\circlearrowleft}}$. Observe that the choice of optional asynchronous collaboration and repeating asynchronous collaboration p_{ac} coincides iff $A_\times = A'_\times$. We similarly extend the respective transitions: $T_{p_{ac},\times} = \{t \in T \mid l(t) \in A_\times \setminus \{\tau\}\} \cup \{t_{p_{ac},\tau} \mid \tau \in A_\times\}$ for $t_{p_{ac},\tau} \notin T$.

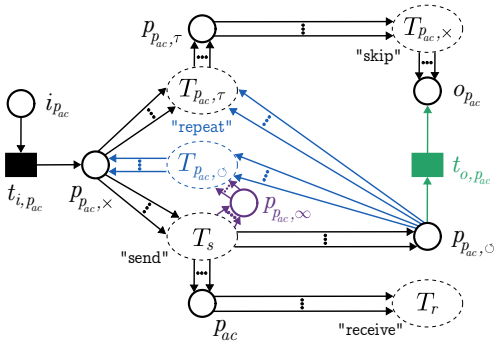

Fig. 4. WF-net $N_{p_{ac}}$ constructed for asynchronous collaboration $(p_{ac}, T_s, T_r) \in AC$.

For each $(p_{ac}, T_s, T_r) \in AC$, construct WF-net $N_{p_{ac}} = (P_{p_{ac}}, T_{p_{ac}}, F_{p_{ac}}, l_{p_{ac}}, \Lambda_\tau)$ as depicted in Fig. 4. The construction copies all transitions $t \in T_s \cup T_r \cup T_{p_{ac},\times} \cup T_{p_{ac},\circlearrowleft}$ as depicted by dashed ovals in Fig. 4. A new transition $t_{i,p_{ac}}$ connects a new source $i_{p_{ac}}$ to a new place $p_{p_{ac},\times} \notin P$ that will be the preset of sending transitions T_s and postset of repeating transitions $T_{p_{ac},\circlearrowleft}$. A set of new τ-labelled transitions $T_{p_{ac},\tau} \cap T = \emptyset$ are added to $N_{p_{ac}}$ to ensure a valid Petri net, if $T_{p_{ac},\times} \neq \emptyset$ (dashed, black oval in Fig. 4). We connect p_{ac} as defined by T_s and T_r by means of the new flow relation $F_{p_{ac}}$ in $N_{p_{ac}}$. Another new place $p_{p_{ac},\circlearrowleft} \notin P$ is the postset of sending transitions and the preset of both a conditional new transition $t_{o,p_{ac}}$ that connects to the new sink place $o_{p_{ac}}$ (green highlighted in Fig. 4) and conditional transitions $T_{p_{ac},\circlearrowleft}$ (blue highlighted "repeat" oval in Fig. 4) to repeat sending of transitions. If $T_{p_{ac},\circlearrowleft} = \emptyset$, no repeating takes place such that the τ-labelled new transition $t_{o,p_{ac}}$ connects $p_{p_{ac},\circlearrowleft}$ with $o_{p_{ac}}$. If $T_{p_{ac},\circlearrowleft} \neq \emptyset$ and the infinite repeating behavior (ii) with unbounded states does not occur for any $m_\circlearrowleft \in M_\circlearrowleft$ of p_{ac}, τ-labelled new transition $t_{o,p_{ac}}$ is replaced by arcs to $T_{p_{ac},\tau}$ and transitions $T_{p_{ac},\circlearrowleft}$. If $T_{p_{ac},\circlearrowleft} \neq \emptyset$ and the infinite repeating behavior (ii) with unbounded states does occur for some $m_\circlearrowleft \in M_\circlearrowleft$ of p_{ac}, then the respective sending activities $T_\infty \subseteq T_s$ and their corresponding repeating activities $T_{\circlearrowleft,\infty} \subseteq T_{p_{ac},\circlearrowleft}$ are additionally connected through the new place $p_{p_{ac},\infty} \notin P$ (purple highlighted in Fig. 4). Place $p_{p_{ac},\infty}$ is only added to $P_{p_{ac}}$, if sending activities fire infinitely often (cf. ii for $m_\circlearrowleft$). Hence, we ensure that sending activities T_∞ in $N_{p_{ac}}$ can fire infinitely often by adding a "token generator" structure to the WF-net, i.e., T_∞ has not only the place p_{ac}, but also the place $p_{p_{ac},\infty}$ as its postset such that through repeating with its corresponding $T_{\circlearrowleft,\infty}$ it can always fire again. By labelling copied transitions in $N_{p_{ac}}$ similar to their original counterparts in cPN and all other new transitions with τ, we encode executing asynchronous collaboration (p_{ac}, T_s, T_r) in newly constructed WF-net $N_{p_{ac}}$.

Let $cc_{p_{ac}} : \{1, \ldots, |P_{AC}|\} \to P_{AC}$ be a concept collection of asynchronous places. Define $cc_{os} = cc \cdot cc_{p_{ac}}$, $WC_{os} = (N_c)_{c \in \mathrm{RNG}(cc_{os})}$ and $CP_{WC_{os}} = (\emptyset, \emptyset, \emptyset, \emptyset, ET_{os})$ with $ET_{os} = \{(t_{sc}, T_{sc}) \subseteq T^u_{os} \times \mathcal{P}^+(T^u_{os}) \mid t_{sc} \in T_{sc} \wedge l^u(t_{sc}) \neq \tau \wedge \forall_{t,t' \in T_{sc}} l^u_{os}(t) = l^u_{os}(t')\}$ the extended set of equally-labelled transitions in the extended workflow collection WC_{os}. Then, $cPN_{os} = \biguplus_{c \in \mathrm{RNG}(cc_{os})}^{CP_{WC_{os}}} N_c$. Define relation $Q \subseteq \mathcal{R}(cPN) \times \mathcal{R}(cPN_{os})$ such that $(m, m') \in Q$ iff $m(p) \leq m'(p)$ for all places $p \in P_{os}$ of cPN_{os}. Then, $cPN \approx_Q cPN_{os}$, since with the exception of new τ-labelled transitions in $N_{p_{ac}}$ for asynchronous collaboration p_{ac}, all transitions of the constructed WF-nets $N_{p_{ac}}$ are fused with their original counterparts in WC, exactly the same flow relation is encoded in $\bullet p_{ac}$ and $p_{ac}\bullet$ as is defined by (4) in Def. 3, and optional and repeating behavior is exactly encoded. By definition only Υ' is contained in cPN_{os}. Since collaboration types $\upsilon_m, \upsilon_h, \upsilon_h$ are defined by $(p_{ac}, T_s, T_r) \in AC$ and $\emptyset$ is the only subset of Υ smaller than Υ', it follows from definition in Def. 10 that Υ' is minimal. ■

In the following, alternative formulation of Theorem 2, we prove the statement by an inefficient construction that copies complete WF-nets of collaboration concepts. Although the alternative formulation does not impose any con-

ditions on the asynchronous collaboration, the size of the constructed cPN_{os} "explodes" quickly.

Theorem (Synchronous activity execution is minimal - Alternative) $\Upsilon' = \{\upsilon_s\}$ *is minimal.*

Proof. Let $cPN = ((P, T, F, l, \Lambda_\tau), m_0) = \biguplus_{c \in \text{RNG}(cc)}^{CP} N_c$ be a collaboration Petri net that contains collaboration types $\upsilon \in \Upsilon$ with $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ a collaboration pattern and $WC = (N_c)_{c \in \text{RNG}(cc)}$ a workflow collection, i.e., $N_c = (P_c, T_c, F_c, l_c, \Lambda_\tau)$ is a WF-net for each $c \in \text{RNG}(cc)$. Define for each $p_{ac} \in P_{AC}$, the set of collaboration concepts $C_{p_{ac}} = \{c \in \text{RNG}(cc) \mid (p_{ac}, T_s, T_r) \in AC \wedge (t \in T_s \vee t \in T_r) \wedge t \in T_c\}$ that asynchronously collaborate via p_{ac}. We construct a new WF-net $N_{p_{ac}}$ for each $(p_{ac}, T_s, T_r) \in AC$ by copying all WF-nets N_c that collaborate via p_{ac}. We set $P_{RS}^{p_{ac}} = P_{RS} \cap \{p_{ac}\}$. The equally-labelled transitions in $N_{p_{ac}}$ are $ET_{p_{ac}} = \{(t_{sc}, T_{sc}) \subseteq T_{p_{ac}} \times \mathcal{P}^+(T_{p_{ac}}) \mid t_{sc} \in T_{sc} \wedge l^u(t_{sc}) \neq \tau \wedge \forall_{t,t' \in T_{sc}} l^u(t) = l^u(t')\}$ with $T_{p_{ac}} = \bigcup_{c \in C_{p_{ac}}} T_c$. Then, $CP_{WC}^{p_{ac}} = (\{p_{ac}\}, P_{RS}^{p_{ac}}, ra_{|\{p_{ac}\}}, \{(p_{ac}, T_s, T_r)\}, ET_{p_{ac}})$. Lastly, $N_{p_{ac}} = \biguplus_{c \in C_{p_{ac}}}^{CP^{p_{ac}}} N_c$.
Let $cc_{p_{ac}} : \{1, \ldots, |P_{AC}|\} \to P_{AC}$ be a concept collection of asynchronous places. Define $cc_{os} = cc \cdot cc_{p_{ac}}$, $WC_{os} = (N_c)_{c \in \text{RNG}(cc_{os})}$ and $CP_{WC_{os}} = (\emptyset, \emptyset, \emptyset, \emptyset, ET_{os})$ with $ET_{os} = \{(t_{sc}, T_{sc}) \subseteq T_{os}^u \times \mathcal{P}^+(T_{os}^u) \mid t_{sc} \in T_{sc} \wedge l^u(t_{sc}) \neq \tau \wedge \forall_{t,t' \in T_{sc}} l_{os}^u(t) = l_{os}^u(t')\}$ the extended set of equally-labelled transitions in the extended workflow collection WC_{os}. Then, $cPN_{os} = \biguplus_{c \in \text{RNG}(cc_{os})}^{CP_{WC_{os}}} N_c$. Define relation $Q \subseteq \mathcal{R}(cPN) \times \mathcal{R}(cPN_{os})$ such that $(m, m') \in Q$ iff $m(p) \leq m'(p)$ for all places $p \in P_{os}$ of cPN_{os}. Then, $cPN \approx_Q cPN_{os}$, since with the exception of new τ-labelled transitions in $N_{p_{ac}}$, all transitions of the constructed WF-nets $N_{p_{ac}}$ are fused with their original counterparts in WC and exactly the same flow relation is encoded in $\bullet p_{ac}$ and $p_{ac}\bullet$ as is defined by (4) in Def. 3. By definition only Υ' is contained in cPN_{os}. Since collaboration types $\upsilon_m, \upsilon_h, \upsilon_h$ are defined by $(p_{ac}, T_s, T_r) \in AC$ and $\emptyset$ is the only subset of Υ smaller than Υ', it follows from definition in Def. 10 that Υ' is minimal. ■

Although $\{\upsilon_s\}$ is only minimal under two conditions for an efficient construction (cf. Theorem 2), both conditions are not realistic in a real-world collaboration process, as the τ-labelled loop condition excludes arbitrary, non-observable sending of messages or requiring of resources and the empty trace would mean that a collaboration process can execute without any observable activity. From Theorem 2, we cannot conclude that $\{\upsilon_s\}$ is the only minimal element of the partial order $\preccurlyeq$. The next theorem proves that synchronous collaboration cannot be simulated by asynchronous collaboration such that $\{\upsilon_s\}$ is the only, unique minimal element of $\preccurlyeq$.

Theorem 3 (Synchronous activity execution cannot be simulated). *Let* $\Upsilon' = \{\upsilon_r, \upsilon_h, \upsilon_m\}$. *Then,* $\Upsilon' \not\preccurlyeq \Upsilon$.

Proof. We prove by contradiction. Assume $\Upsilon' \preccurlyeq \Upsilon$. Let $cPN = \biguplus_{c \in \text{RNG}(cc)}^{CP} N_c$ with $CP_{WC} = (P_{AC}, P_{RS}, ra, AC, ET)$ be a collaboration Petri net that contains

all $\upsilon \in \Upsilon$ and $cPN_{sc} = \biguplus^{CP}_{c \in \mathrm{RNG}(cc_{sc})} N_c$ with $CP_{WC_{sc}} = (P_{AC,sc}, P_{RS,sc}, ra_{sc}, AC_{sc}, \emptyset)$ be a collaboration Petri net that contains only $\upsilon' \in \Upsilon'$ such that $cPN \approx cPN_{sc}$. Observe that all three synchronous collaboration types are represented by AC and do not merge transitions or change their label in Def. 3. Hence, from Def. 10 it follows that all transitions $t \in T_{sc}, (t_{sc}, T_{sc}) \in ET$ are still present in WC_{sc} (changing the workflow collection WC is prohibited) such that $CP_{WC_{sc}} = (P_{AC,sc}, P_{RS,sc}, ra_{sc}, AC_{sc}, ET)$ and $\emptyset \neq ET$[4]. ■

Overall, our theory on collaboration types in collaboration Petri nets demonstrates that modulo weak bisimilarity, asynchronous collaboration can be simulated by synchronous collaboration, but not vice versa. The implications are twofold. First, it suffices to prove weak bisimilarity of discovered Petri nets by some CPD technique *discover* for υ_s in the next section, if *discover* constructs similar Petri net patterns for the collaboration types. Second, if a CPD technique *discover* can only mine asynchronous collaboration types, it is not possible to transform the input cL' or tweak *discover* in such a way that it can mine synchronous collaborations.

4.3 Bisimulation of *cPN* and Discovered Petri Nets

This section shows that for any discovered model $M = discover(cL')$ there exists a collaboration Petri net cPN that is weakly bisimilar to M, if $Q_{discover}$ holds (cf. Fig. 3). $Q_{discover}$ holds iff (i) the discovered model M is a labelled Petri net N as defined in this paper (cf. Sect. 2), (ii) only supports collaboration types Υ with corresponding Petri net patterns that are weakly bisimilar to the typical patterns in Fig. 2 and Def. 5, and (iii) takes the divide-and-conquer approach as depicted in Fig. 3 with some custom model composition *comp* and some custom mining functionality *cdisc* to mine asynchronous collaborations (synchronous collaborations are always determined by equal labels). The third condition in $Q_{discover}$ means that *discover* projects the input log cL' to logs $L_{cc(1)}, \ldots, L_{cc(n)}$, applies process discovery *disc* on each log $L_{cc(1)}, \ldots, L_{cc(n)}$, mines some Petri net N_{ac} to represent the asynchronous collaborations, and composes with some custom model composition $comp(N_{cc(1)}, \ldots, N_{cc(n)}, N_{ac}) = N$.

To illustrate property $Q_{discover}$, the CPD techniques [15] to discover typed Jackson nets and OCPD [8] to discover OCPNs (cf. Tab. 1 and Sect. 5) do not have property $Q_{discover}$, as their discovered models are defined with multiple higher-level net concepts, e.g., token colors, that violate (i). However, OCPD discovers a labelled Petri net N after "step 3" that is a cPN without the global source and sink place. Only in the next steps, N is extended to OCPN with *place*

[4] Although the contradiction follows from a technicality, the statement still holds, if we enable synchronizing arbitrary transitions in a collaboration pattern CP in Def. 3, which is rarely done. To the best of our knowledge, there exists only a single modeling technique [5] that enables synchronizing arbitrary transitions. Nevertheless, asynchronous collaboration does not merge transitions, so we can never avoid firing traces σ that include all the labels of synchronized transitions. These labels cannot be simulated by a single, synchronizing label.

types, token colors and variable arcs. Similarly, their implementation in PM4PY[5] internally represents OCPN as labelled Petri net N that is extended with the higher-level net concepts for visualization and export only. Besides, for one-to-one correspondences between the collaboration concept's cases, the higher-level net concepts in an OCPN are not required. Hence, for our collaboration event log cL, OCPD has property $Q_{discover}$, which we demonstrate in the next section.

Two exemplary CPD techniques from Tab. 1 that have $Q_{discover}$ are *cross-department collaborative healthcare process* (CCPH) discovery [53] and the *Colliery* technique [24]. For the sake of brevity, we only report the reasons for CCHP in detail. CCHP discovers composed RM_WF_nets that are labelled[6] Petri nets (cf. Def. 5 and Def. 9 in [53]), i.e., (i) satisfied. CCHP discovers all four collaboration types $\upsilon \in \Upsilon$ using exactly the same Petri net patterns as depicted in Fig. 2 (cf. Definition 6, 7, and 8 in [53]), i.e., (ii) is satisfied. Given a collaboration event log cL' as a XES log (cf. *convert* in Sect. 4.1), CCHP works exactly as required by (iii): The collaboration event log cL' is projected onto each department's event log $L_{cc(1)}, \ldots, L_{cc(n)}$ for cc a concept collection of n departments in a hospital. For each department's event log $L_{cc(i)}, i \in \{1, \ldots, n\}$, a WF-net $N_{cc(i)}$ is discovered using process discovery technique *disc* (cf. Line 2 of Algorithm 1 in [53]). Then, Petri net N_{ac} representing all collaborations between departments discovered by $cdisc_{CCHP}$ (cf. Algorithm 2 in [53]), is composed with all WF-nets $N_{cc(i)}$ to yield the composed RM_WF_nets N. Hence, (iii) is satisfied and CCHP has property $Q_{discover}$.

Note that the construction to create a WF-net for each message channel and resource type $p_{ac} \in P_{AC}$ used to prove that the three asynchronous collaboration types υ_m, υ_h, and υ_r are simulated by synchronous activity execution υ_s (cf. Theorem 2) can also be applied to CPD technique *discover* that has property $Q_{discover}$, in particular (ii). From Theorem 2, we can simulate any N_{ac} with asynchronous collaborations by synchronous collaborations, i.e., equally-labelled transitions in the WF-nets $N_{cc(1)}, \ldots N_{cc(n)}$ and additional WF-nets $N_{p_{ac}}$ for each asynchronous collaboration p_{ac} in N_{ac}. Hence, the custom model composition *comp* in *discover* reduces to the following model composition:

Definition 11 (Model Composition [8,53]). *Let $WC = (N_c)_{c \in \mathrm{RNG}(cc)}$ be a workflow collection. The model composition is a marked Petri net $(N, m_0) = \bigcup_{c \in \mathrm{RNG}(cc)} N_c$ with $N = (P, T, F, l, \Lambda_\tau)$ defined by:*

- *$P = P^u$ as defined in Def. 2,*
- *$T = \bigcup_{c \in \mathrm{RNG}(cc)} r(T_c)$, with r a renaming function: $r(x) = t_s$ if there exists $T_s \in ST$ such that $x \in T_s$ and $t_s \in T_s$ a fixed transition with $ST = \{T_s \mid T_s \subseteq T^u \wedge \forall_{t,t' \in T_s} l^u(t) = l^u(t') \wedge l^u(t) \neq \tau\}$ the set of equally-labelled (synchronous) transition subsets, otherwise $r(x) = x$,*

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

[6] [53] does not define labelling l. Without labelling, WF-nets with different τ-labelled transitions are impossible. WF-nets with different τ-labelled transitions are discovered by many process discovery techniques, e.g., *Inductive* [52] or *Split Miner* [14]. CCHP uses the Inductive Miner and Split Miner [53], i.e., we assume labelling l.

- $F = \{(r(x), r(y)) \mid (x, y) \in F^u\}$,
- $l = l^u$,
- $m_0(p) = [i_{cc(1)}, \ldots, i_{cc(n)}]$.

Given collaboration event log cL that contains collaboration concepts in cc and a CPD technique $discover$ with property $Q_{discover}$, we define $discover(cL') = \bigcup_{c \in \text{RNG}(cc')}(disc(\pi_c(cL')))_{c \in \text{RNG}(cc)} \cdot (N_c)_{c \in \text{RNG}(cc_{ac})}$[7] with $cc' = cc \cdot cc_{ac}$ for $(N_c)_{c \in \text{RNG}(cc_{ac})}$ the workflow collection of WF-nets constructed to simulate asynchronous collaboration by synchronous collaboration. Coming back to the discovery framework in Fig. 3, we are left to prove that $discover(cL') = N \approx cPN$.

Theorem 4 (Composed models are weakly bisimilar to collaboration Petri nets). *Let $(N, m_0) = \bigcup_{c \in \text{RNG}(cc)} N_c$ be a model composition. Then, there exists collaboration Petri net cPN such that $N \approx cPN$.*

Proof. Given $(N, m_0) = \bigcup_{c \in \text{RNG}(cc)} N_c$ with $N = (P, T, F, l, \Lambda_\tau)$ for workflow collection $WC = (N_c)_{c \in \text{RNG}(cc)}$. Define collaboration pattern $CP_{WC} = (\emptyset, \emptyset, \emptyset, \emptyset, ET)$ with ET a set that satisfies property (6) in Def. 3 and $|ET| = |ST|$ as defined in Def. 11. From Def. 3 (6), it follows that $ST = \{T_{sc} \mid (t_{sc}, T_{sc}) \in ET\}$. Then, $cPN = (N', m_0') = \biguplus_{c \in \text{RNG}(cc)} N_c$. The statement follows from Def. 4, Def. 11, and from relation $Q \subseteq \mathcal{R}(N, m_0) \times \mathcal{R}(N', m_0')$ defined by $(m, m') \in Q$ iff $m(p) \leq m'(p)$ for all places $p \in P$ of the model composition's net N. ■

Consequently, for a CPD technique $discover$ that has property $Q_{discover}$, every discovered Petri net $N = discover(cL')$ is weakly bisimilar to a collaboration Petri net cPN. As our proofs are constructive, we are always able to explicitly construct, export or visualize the weakly bisimilar collaboration Petri net such that subsequent analysis techniques in a collaboration mining pipeline can be developed for and applied on collaboration Petri nets. In the next section, we demonstrate how collaboration Petri nets allow to compare CPD techniques.

5 Comparative Evaluation

In this section, we present results from comparing CPD techniques with our discovery framework (cf. Fig. 3). There exist at least 14 techniques $discover$ as depicted in Tab. 1. Each technique discovers a specific class of models $N = discover(cL')$. The class of discovered models can either be defined using *basic* net concepts as introduced in this paper (cf. Sect. 2) or using *higher*-level net concepts such as token colors, variable arcs, and place types (cf. Sect. 3.2).

From the fourteen CPD techniques in Tab. 1, only four are publicly available such that we can apply the implementation in our publicly available[8] framework implementation. The four publicly available CPD techniques are CCHP [53],

[7] Concatenation is similarly defined for tuples as for traces.
[8] https://github.com/janikbenzin/cpn_discovery

Table 1. Overview of existing CPD techniques *discover*.

CPD	Year	Model	Υ	Domains	Net Concepts
[60,65]	2013/15	Artifact-centric models	υ_s	Accounting	Basic
[8]	2020	Object-centric Petri nets	υ_s	Commerce	Higher
[15]	2023	Typed Jackson nets	υ_s	Commerce	Higher
[40]	2008	WF-nets	υ_m	Web service	Basic
[69]	2019	Communication nets	υ_m	Web service	Basic
[39]	2022	System net	$\upsilon_s, \upsilon_m, \upsilon_h$	Retail	Higher
[59]	2023	Generalized WF-nets	υ_s, υ_m	Multi-agent systems	Basic
[72]	2023	Multi-agent system net	υ_h	Healthcare & other	Basic
[77]	2013	Integrated RM_WF_nets	$\upsilon_m, \upsilon_s, \upsilon_r$	Logistics, Healthcare	Basic
[75]	2020	Top-level process model	υ_m	Logistics	Basic
[24]	2022	BPMN collab. diagram	υ_m	Healthcare & other	Basic
[50]	2022	Industry net	υ_m	Theoretical	Basic
[53]	2023	Composed RM_WF_nets	Υ	Healthcare	Basic

Colliery [24], OCPD [8], and Agent Miner [72]. Although OCPD generally uses higher-level net concepts, for our collaboration event logs cL (cf. Def. 7) OCPD discovers models N that are weakly bisimilar to collaboration Petri nets in an intermediate step of the technique as explained in more detail in Sect. 4.3. Taking the intermediate representation for OCPD into account, all of the four CPD techniques have the property $Q_{disocver}$ and use the Inductive Miner for *disc*.

We evaluate in a controlled setting similar to [74] in which the *true* cPN that generates cL is known. As the generation of various collaboration Petri nets cPN is out of scope, an existing dataset consisting of multiple cPN and cL is selected. Selection criteria are a diverse set of collaboration types and collaboration patterns while maintaining a majority of the four CPD techniques to be applicable. The dataset in [58] covers υ_m and υ_s in twelve collaboration Petri nets each with a distinct collaboration pattern. The following datasets are not selected: [53], [75], and logs from the *Business Process Intelligence Challenge* as organized by the IEEE Task Force on Process Mining[9] do not feature a true collaboration Petri net, [24] only covers υ_m, and public OCEL logs[10] only cover υ_s. Our selected dataset means that Agent Miner is inapplicable, as Agent Miner can only discover υ_h that is a special case of υ_m (cf. Sect. 3.1).

Our selected dataset models twelve multi-agent systems consisting of two or three agents ($|cc| = 2$ or $|cc| = 3$ in Tab. 2) whose collaboration patterns represent various (service) interaction patterns as specified in [16] and described in [59]. Collaboration patterns vary along the number of message *transmissions* via channel p_{ac} (*trans* in Tab. 2), the number of message exchanges classified into one-way, two-way or multiple exchanges (*# col.*), and the relation one-to-one (denoted by 1:1), one-to-many (1:n), and many-to-many (m:n) between sending and receiving activities of the agents via channel p_{ac} (cf. *act. rel.*). For each cPN, the dataset contains a collaboration event log cL with $|cL| = 5000$ traces generated from cPN. Further metrics of cL are reported in Tab. 2.

[9] https://www.tf-pm.org/competitions-awards/bpi-challenge
[10] https://www.ocel-standard.org/event-logs/overview/

For each cL, we apply *convert* resulting in a collaboration event log cL' that is in the format required by the respective CPD technique. Note that we can "apply" the simulation of collaboration type υ_m by υ_s also during conversion to cL', i.e., the OCEL cL' for OCPD includes message object types such that υ_m is simulated by υ_s on the log-level (cf. Sect. 4.1), but in a version that does not properly simulate optional and repeated asynchronous collaboration. The implementation of a construction that completely mirrors Theorem 2 on the log-level is left to future work (cf. future work in Sect. 7). For CCHP and Colliery simulation of types is not required, as they discover υ_m (cf. Fig. 2). Due to Theorem 3, the reverse simulation is not possible, i.e., Colliery cannot discover υ_s such that *fitness* is reduced.

Table 2. Comparing CPD techniques CCHP [53], Colliery [24], and OCPD [8] with our discovery framework to the true cPN that generated cL [58,59].

Col.	event log cL	1	2	3	4	5	6	7	8	9	10	11	12
	Events	95052	149988	92668	102404	182452	123322	88068	157098	115000	102548	160000	88089
	$\lvert cc\rvert$	2	2	2	2	2	2	2	3	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 : trans.	single	single	single	single	single	single	multi	multi	single	single	single	single
	υ_m: # col.	one	one	one	two	two	two	two	multi	two	two	two	two
	υ_m: act. 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
	Min. trace	17	29	17	18	36	24	8	30	23	20	32	17
	Avg. trace	19	30	19	20	36	25	18	31	23	21	32	18
	Max. trace	21	31	20	23	37	25	29	32	23	21	32	18
True	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
	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
	Sound	yes	no	no	no	no	yes	yes	yes	yes	yes	yes	yes
CCHP	Fitness	ex	ex	**1.0**	ex	ex	ex	ex	ex	ex	ex	ex	0.384
	Precision	ex	ex	**0.765**	ex	ex	ex	ex	ex	ex	ex	ex	0.583
	Sound	no	no	no	no	no	no	no	no	no	no	no	no
Colliery	Fitness	0.863	0.966	1.0	0.683	0.965	**0.926**	0.587	**0.699**	0.64	0.711	0.93	**0.893**
	Precision	0.722	**0.426**	**0.765**	0.776	**0.26**	**0.597**	0.687	**0.306**	0.5	0.669	**0.468**	0.697
	Sound	no	no	no	no	no	no	no	no	no	no	no	no
OCPD	Fitness	**1.0**	**1.0**	ex	**1.0**	**1.0**	ex	**0.727**	ex	**1.0**	**1.0**	**1.0**	0.818
	Precision	**0.876**	0.351	ex	**0.792**	0.156	ex	**0.956**	ex	**0.892**	**0.919**	0.431	**0.887**
	Sound	yes	no	no	yes	no	no	no	no	yes	no	no	no

To evaluate $discover(cL') = N$, we construct cPN' with $N \approx cPN'$ (cf. Sect. 4.3) and apply *alignment-based fitness* [4] and *alignment-based precision* [12] as implemented in PM4PY. Although [63] propose monotone alternatives to both alignment-based metrics, their computation with *Entropia*[11] on cPN' resulted in >16 GB heap space and execution times >3 hours (more than [13] allowed for their benchmark study), so we chose the more efficient alignment-based metrics. With PM4PY's soundness verification, we also report its results. Additionally, we compute fitness, precision and soundness for the true collaboration Petri net cPN from the dataset (*True* in Tab. 2).

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

As CCHP assumes a unique message channel p_{ac} per sending/receiving activity pair (1:1) and adds each channel with their respective arcs to the discovered model cPN', their construction often results in a cPN' with $[o] \notin \mathcal{R}(cPN')$, if *act. rel* of υ_m is 1:n or m:n. Without $[o] \in \mathcal{R}(cPN')$, alignments fail, which we report as *ex* in Tab. 2. Although Colliery assumes 1:1 for *act. rel.*, it picks one of the unique pairs and adds a message channel for this pair only, i.e., the construction always results in $[o] \in \mathcal{R}(cPN')$ for the twelve cL, but with a reduced precision. Additionally, Colliery does not discover a sound model given any of the twelve logs cL. However, four of the twelve true models are not sound.

OCPD is the only CPD technique that discovers some sound models. Three times, OCPD discovers cPN' with $[o] \notin \mathcal{R}(cPN')$, because there is a choice between sending message via two channels p_{ac} and p'_{ac}. Our currently implemented construction to simulate asynchronous collaboration on the log-level does not allow *exclusive* channels, i.e., two optional channels $(p_{ac}, T_s, T_r), (p'_{ac}, T'_s, T'_r) \in AC$ and no trace $\sigma \in cL$ with $m_0 \xRightarrow{\sigma} [o]$ in the true cPN exists such that $\exists_{i,j \in \{1,\dots,|\sigma|\}} \sigma(i) \in l^u(T_s) \wedge \sigma(j) \in l^u(T'_s)$. As the implementation is missing the skipping of optional asynchronous collaboration, the collaboration Petri net of OCPD never reaches marking $[o]$ in case of exclusive channels (cf. future work in Sect. 7). Hence, no alignment is possible. If there are no exclusive channels, OCPD often discovers the most fitting and the most precise models. Surprisingly, for one of the unsound true cPN, OCPD discovers a sound model that is not only as fitting as the true model, but even more precise.

To sum up, CCHP is sensitive to the relationship between sending and receiving activities per message channel. Colliery is the most robust technique, as it always discovers a relatively fitting and precise model. OCPD is less robust than Colliery due to exclusive channels, but discovers the most fitting models.

6 Related Work

First, [17] gives a recent overview and analysis of existing Petri net classes to model collaboration processes that are alternatives to collaboration Petri nets. [17] concludes that existing work in modeling of collaboration processes spans many research areas (cf. Sect. 3.1) and is focused on a specific domain or type of collaboration. Furthermore, multiple Petri net extensions are used such that existing techniques are not applicable anymore, while no increased modeling power is achieved (cf. [77,76,53]). An important distinction that holds for any Petri net class is given in Sect. 5 and depicted for the models of CPD techniques in Tab. 1: Basic vs. higher-level net concepts. Many classes with higher-level net concepts exist [35,64], but for many collaboration processes these concepts are not required (cf. Sect. 3.2). Collaboration Petri nets resemble characteristics of the class in [76] as it is the first class with all four collaboration types and of the class in [5] (cf. Sect. 3.2). Both classes are integrated into a single class and extended with respect to their respective advantageous properties such as collaboration types and conciseness. Also, a consistent labelling with silent ac-

tions is introduced to formalize activities similar to [8] and to state a Petri net's semantics in an activity-centric manner.

Second, CPD techniques as depicted in Tab. 1 are to the best of our knowledge not compared yet, because a standard with a theoretical foundation that enables comparability was missing. Typically, CPD techniques compare with process discovery techniques applied on the collaborating case [77,72,53] or do not compare with any other discovery technique, but only via varying the technique-specific parameters [69,24]. Third, process discovery techniques are extensively compared in [27,23,20,13]. [13] is the most recent benchmark and does not need to develop a theory to standardize Petri net classes to model process orchestrations, as WF-nets were studied before and have emerged as the de-facto standard already. In that regard, we study collaboration Petri net with the aim of bringing CPD research to the same standardization as process discovery research already experiences. Fourth, collaboration event logs can be generated from other generating models such as web service compositions [32,31], *inter-organization business processes* [33], multi-agent systems [47,57], *communicating resource systems* [69], and process choreographies [18]. Despite the heterogeneity, we generate collaboration event logs given a WF-net, i.e., the collaboration Petri net, and the collaboration labelling. Hence, in the context of our representational bias (cf. Sect. 3.2), it is possible to apply existing event log generators for WF-nets [26,21] with minor adjustments on the labelling.

7 Conclusion and Outlook

In this paper, we presented collaboration Petri nets as a standard Petri net class for modeling and discovery of collaboration processes along with formal and empirical evidence to show that collaboration Petri nets meet the standard's requirements in a balanced way. Modeling power is sufficient to represent most collaboration processes that feature collaboration types Υ and occur in the domains healthcare, web services, logistics, and multi-agent systems. While collaboration processes with *1:n* or *m:n* relationships between it's collaboration concepts cases can only be represented with extensions, collaboration Petri nets with their WF-net structure benefit from decidability of (conventional) soundness, recursive applicability to deal with various granularity levels, and the vast body of knowledge, techniques, and tools available for WF-nets. Due to the syntactic sugar for the asynchronous collaboration types, the sole reliance on well-known, basic net concepts, and constructive proofs that enable comparability between CPD techniques in our discovery framework, modeling convenience is adequate. If a CPD technique *discover* has property $Q_{discover}$, it discovers labelled Petri nets N that are weakly bisimilar to collaboration Petri net. Since at least the three existing CPD techniques CCHP, Colliery, and OCPD have property $Q_{discover}$, collaboration Petri net posses desirable relations to existing model classes.

Outlook. Simulating asynchronous collaboration results in collaboration Petri nets and logs that may present technical difficulties for empirical evaluation, so

we will propose a refined construction in our implementation that is not only weakly bisimilar, but preserves reaching the global sink place as is theoretically shown in Theorem 2. A more thorough, comparative maturity analysis of more existing CPD techniques that possess $Q_{discover}$ with our discovery framework brings collaboration mining closer to the state of process mining. Furthermore, the framework in [15] achieves *rediscoverability* of typed Jackson nets by projecting onto the subset-closed set of type combinations. Their construction indicates that our framework in Fig. 3 has to be extended to project onto the subset-closed set of collaboration concept combinations to guarantee *rediscoverability* of collaboration Petri nets. Under what circumstances can we transfer the theory in [15] to collaboration Petri nets remains to be investigated.

References

1. IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. IEEE Std 1849-2016 pp. 1–50 (Nov 2016). https://doi.org/10.1109/IEEESTD.2016.7740858, https://ieeexplore.ieee.org/document/7740858, conference Name: IEEE Std 1849-2016
2. Aalst, W.M.P.: Verification of workflow nets. In: Goos, G., Hartmanis, J., Leeuwen, J., Azéma, P., Balbo, G. (eds.) Application and Theory of Petri Nets 1997, vol. 1248, pp. 407–426. Springer Berlin Heidelberg, Berlin, Heidelberg (1997), http://link.springer.com/10.1007/3-540-63139-9_48, series Title: Lecture Notes in Computer Science
3. van der Aalst, W.M.P.: On the Representational Bias in Process Mining. In: 2011 IEEE WETICE. pp. 2–7 (Jun 2011)
4. 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). https://doi.org/10.1002/widm.1045, https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1045,
5. van der Aalst, W.M.P.: Modeling and analyzing interorganizational workflows. In: Proceedings 1998 ACSD. pp. 262–272 (1998)
6. van der Aalst, W.M.P.: Interorganizational workflows: An approach based on message sequence charts and petri nets. Systems Analysis Modelling Simulation **34**, 335–367 (1999)
7. van der Aalst, W.M.P.: Service Mining: Using Process Mining to Discover, Check, and Improve Service Behavior. IEEE Trans. Serv. Comput. **6**(4), 525–535 (Oct 2013)
8. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundamenta informaticae **175**(1-4), 1–40 (2020)
9. van der Aalst, W.M.P., van Hee, K.M., Massuthe, P., et al., S.: Compositional Service Trees. In: Applications and Theory of Petri Nets. pp. 283–302. LNCS, Springer, Berlin, Heidelberg (2009)
10. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans Knowl Data Eng **16**(9), 1128–1142 (Sep 2004)
11. van der Aalst, W.M.P., Weske, M.: The P2P Approach to Interorganizational Workflows. In: Advanced Information Systems Engineering. pp. 140–156. LNCS, Springer, Berlin, Heidelberg (2001)

12. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Measuring precision of modeled behavior. Information Systems and e-Business Management **13**(1), 37–67 (Feb 2015). https://doi.org/10.1007/s10257-014-0234-7, https://doi.org/10.1007/s10257-014-0234-7
13. Augusto, A., Conforti, R., Dumas, M., Rosa et al., M.L.: Automated Discovery of Process Models from Event Logs: Review and Benchmark. IEEE Trans Knowl Data Eng **31**(4), 686–705 (Apr 2019)
14. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems **59**(2), 251–284 (May 2019). https://doi.org/10.1007/s10115-018-1214-x, https://doi.org/10.1007/s10115-018-1214-x
15. Barenholz, D., Montali, M., Polyvyanyy, A., Reijers et al., H.A.: There and Back Again. In: PETRI NETS 2023. pp. 37–58. LNCS, Springer, Cham (2023)
16. Barros, A., Dumas, M., ter Hofstede, A.H.M.: Service Interaction Patterns. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) Business Process Management. pp. 302–318. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2005)
17. Benzin, J.V., Rinderle-Ma, S.: Petri Net Classes for Collaboration Mining: Assessment and Design Guidelines (Sep 2023), arXiv:2309.06200 [cs], Accepted by ICPM Workshops 2023.
18. Bischoff, F., Fdhila, W., Rinderle-Ma, S.: Generation and Transformation of Compliant Process Collaboration Models to BPMN. In: Giorgini, P., Weber, B. (eds.) Advanced Information Systems Engineering. pp. 462–478. Lecture Notes in Computer Science, Springer International Publishing, Cham (2019)
19. Borkowski, M., Fdhila, W., Nardelli, M., Rinderle-Ma, S., Schulte, S.: Event-based failure prediction in distributed business processes. Inf. Syst. **81**, 220–235 (2019)
20. vanden Broucke, S.K., De Weerdt, J., Vanthienen, J., Baesens, B.: A comprehensive benchmarking framework (cobefra) for conformance analysis between procedural process models and event logs in prom. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp. 254–261. IEEE (2013)
21. Burattin, A., Sperduti, A.: Plg: A framework for the generation of business process models and their execution logs. In: Business Process Management Workshops: BPM 2010 International Workshops and Education Track, Hoboken, NJ, USA, September 13-15, 2010, Revised Selected Papers 8. pp. 214–219. Springer (2011)
22. Chu, X.N., Tso, S.K., Zhang, W.J., Li, Q.: Partnership Synthesis for Virtual Enterprises. Int. J. Adv. Manuf. Technol. **19**(5), 384–391 (Mar 2002)
23. Claes, J., Poels, G.: Process mining and the prom framework: an exploratory survey. In: Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers 10. pp. 187–198. Springer (2013)
24. Corradini, F., Re, B., Rossi, L., Tiezzi, F.: A Technique for Collaboration Discovery. In: Enterprise, Business-Process and Information Systems Modeling. pp. 63–78. LNBIP, Springer, Cham (2022)
25. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do practitioners use conceptual modeling in practice? Data & Knowledge Engineering **58**(3), 358–380 (Sep 2006)
26. De Medeiros, A.A., Günther, C.W.: Process mining: Using cpn tools to create test logs for mining algorithms. In: Proceedings of the sixth workshop on the practical use of coloured Petri nets and CPN tools (CPN 2005). pp. 177–190. University of Aarhus (2005)

27. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. Information systems **37**(7), 654–676 (2012)
28. Decker, G., Weske, M.: Local Enforceability in Interaction Petri Nets. In: BPM 2007, vol. 4714, pp. 305–319. Springer, Berlin, Heidelberg (2007)
29. Decker, G., Weske, M.: Interaction-centric modeling of process choreographies. Inf. Syst. **36**(2), 292–312 (Apr 2011)
30. Diba, K., Batoulis, K., Weidlich, M., Weske, M.: Extraction, correlation, and abstraction of event data for process mining. WIREs Data Mining and Knowledge Discovery **10**(3), e1346 (2020)
31. Dustdar, S., Gombotz, R.: Discovering web service workflows using web services interaction mining. International Journal of Business Process Integration and Management **1**(4), 256 (2006). https://doi.org/10.1504/IJBPIM.2006.012624, http://www.inderscience.com/link.php?id=12624
32. Dustdar, S., Gombotz, R., Baïna, K.: Web services interaction mining. Tech. rep., Technical Report TUV-1841-2004-16, Information Systems Institute, Vienna ... (2004)
33. Engel, R., Krathu, W., Zapletal, M., Pichler, C., Bose, R.P.J.C., van der Aalst, W., Werthner, H., Huemer, C.: Analyzing inter-organizational business processes. Information Systems and e-Business Management **14**(3), 577–612 (Aug 2016)
34. Esparza, J., Nielsen, M.: Decidability Issues for Petri Nets. BRICS Report Series. **1**(8), (May 1994)
35. Fahland, D.: Describing Behavior of Processes with Many-to-Many Interactions. In: Application and Theory of Petri Nets and Concurrency. pp. 3–24. LNCS, Springer, Cham (2019)
36. Farshidi, S., Kwantes, I.B., Jansen, S.: Business process modeling language selection for research modelers. Software and Systems Modeling (May 2023). https://doi.org/10.1007/s10270-023-01110-8, https://doi.org/10.1007/s10270-023-01110-8
37. Fdhila, W., Indiono, C., Rinderle-Ma, S., Reichert, M.: Dealing with change in process choreographies: Design and implementation of propagation algorithms. Inf. Syst. **49**, 1–24 (Apr 2015)
38. Fdhila, W., Rinderle-Ma, S., Knuplesch, D., Reichert, M.: Change and Compliance in Collaborative Processes. In: 2015 IEEE SCC. pp. 162–169 (Jun 2015)
39. Fettke, P., Reisig, W.: Systems Mining with Heraklit: The Next Step (Jun 2022), arXiv:2202.01289 [cs]
40. Gaaloul, W., Baïna, K., Godart, C.: Log-based mining techniques applied to Web service composition reengineering. Serv. Oriented Comput. Appl. **2**(2), 93–110 (Jul 2008)
41. Gaaloul, W., Bhiri, S., Godart, C.: Research challenges and opportunities in web services mining. Proc of System and Information Service Web, INFORSID2006 (2006)
42. Garcia, C.d.S., Meincheim, A., Faria Junior, E.R., Dallagassa, M.R., Sato, D.M.V., Carvalho, D.R., Santos, E.A.P., Scalabrin, E.E.: Process mining techniques and applications – A systematic mapping study. Expert Systems with Applications **133**, 260–295 (Nov 2019). https://doi.org/10.1016/j.eswa.2019.05.003, https://www.sciencedirect.com/science/article/pii/S0957417419303161
43. Garcia, E., Giret, A., Botti, V.: Designing normative open virtual enterprises. Enterp. Inf. Syst. **10**(3), 303–324 (Mar 2016)

44. Grefen, P., Mehandjiev, N., Kouvas, G., Weichhart et al., G.: Dynamic business network process management in instant virtual enterprises. Comput. Ind. **60**(2), 86–103 (Feb 2009)
45. Havey, M.: Essential business process modeling. " O'Reilly Media, Inc." (2005)
46. Indulska, M., Green, P., Recker, J., Rosemann, M.: Business Process Modeling: Perceived Benefits. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) Conceptual Modeling - ER 2009. pp. 458–471. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2009)
47. Ito, S., Vymětal, D., Šperka, R., Halaška, M.: Process mining of a multi-agent business simulator. Computational and Mathematical Organization Theory **24**(4), 500–531 (Dec 2018)
48. Jablonski, S., Bussler, C.: Workflow management: modeling concepts, architecture and implementation. ITP New Media (1996)
49. Jensen, K.: Coloured petri nets and the invariant-method. Theor. Comput. Sci. **14**(3), 317–336 (1981)
50. Kwantes, P., Kleijn, J.: Distributed Synthesis of Asynchronously Communicating Distributed Process Models. In: Transactions on Petri Nets and Other Models of Concurrency XVI, pp. 49–72. LNCS, Springer, Berlin, Heidelberg (2022)
51. Kwantes, P.M., Kleijn, J.: On the synthesis of industry level process models from enterprise level process models. In: ATAED@ Petri Nets/ACSD. pp. 6–22 (2018)
52. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In: Colom, J.M., Desel, J. (eds.) Application and Theory of Petri Nets and Concurrency. pp. 311–329. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2013)
53. Liu, C., Li, H., Zhang, S., Cheng et al., L.: Cross-Department Collaborative Healthcare Process Model Discovery From Event Logs. IEEE Trans. Autom. Sci. Eng. **20**(3), 2115–2125 (Jul 2023)
54. Mahulea, C., Mahulea, L., García Soriano, J.M., Colom, J.M.: Modular Petri net modeling of healthcare systems. Flex. Serv. Manuf. J. **30**(1), 329–357 (Jun 2018)
55. Meyer, A., Pufahl, L., Batoulis, K., Fahland, D., Weske, M.: Automating data exchange in process choreographies. Inf. Syst. **53**, 296–329 (Oct 2015)
56. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4), 541–580 (1989), publisher: IEEE
57. Nesterov, R.A., Mitsyuk, A.A., Lomazova, I.A.: Simulating Behavior of Multi-Agent Systems with Acyclic Interactions of Agents. Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS) **30**(3), 285–302 (2018), https://ispranproceedings.elpub.ru/jour/article/view/536, number: 3
58. Nesterov, R.: Compositional discovery of architecture-aware and sound process models from event logs of multi-agent systems: experimental data. (May 2021). https://doi.org/10.5281/zenodo.5830863, https://zenodo.org/records/5830863
59. 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)
60. Nooijen, E.H.J., van Dongen, B.F., Fahland, D.: Automatic Discovery of Data-Centric and Artifact-Centric Processes. In: La Rosa, M., Soffer, P. (eds.) Business Process Management Workshops. pp. 316–327. LNBIP, Springer, Berlin, Heidelberg (2013)
61. Peterson, J.L.: Petri Nets. ACM Computing Surveys **9**(3), 223–252 (Sep 1977)
62. Peterson, J.L.: Petri net theory and the modeling of systems. Prentice Hall PTR (1981)

63. Polyvyanyy, A., Kalenkova, A.: Monotone Conformance Checking for Partially Matching Designed and Observed Processes. In: 2019 International Conference on Process Mining (ICPM). pp. 81–88 (Jun 2019)
64. Polyvyanyy, A., van der Werf, J.M.E.M., Overbeek, S., Brouwers, R.: Information Systems Modeling: Language, Verification, and Tool Support. In: Advanced Information Systems Engineering. pp. 194–212. LNCS, Springer, Cham (2019)
65. Popova, V., Fahland, D., Dumas, M.: Artifact Lifecycle Discovery. International Journal of Cooperative Information Systems **24**(01), 1550001 (Mar 2015)
66. Rinderle-Ma, S., Reichert, M., Jurisch, M.: Equivalence of Web Services in Process-Aware Service Compositions. In: 2009 IEEE ICWS. pp. 501–508 (Jul 2009)
67. Salah, M., Mancoridis, S.: Toward an environment for comprehending distributed systems. In: 10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings. pp. 238–247. IEEE, Victoria, BC, Canada (2003)
68. Schulte, S., Hoenisch, P., Hochreiner, C., Dustdar et al., S.: Towards Process Support for Cloud Manufacturing. In: 2014 IEEE 18th EDOC. pp. 142–149 (Sep 2014)
69. Stroiński, A., Dwornikowski, D., Brzeziński, J.: A Distributed Discovery of Communicating Resource Systems Models. IEEE Trans. Serv. Comput. **12**(2), 172–185 (Mar 2019)
70. Tan, W., Xu, W., Yang, F., Xu et al., L.: A framework for service enterprise workflow simulation with multi-agents cooperation. Enterp. Inf. Syst. **7**(4), 523–542 (2013)
71. Tour, A., Polyvyanyy, A., Kalenkova, A.: Agent System Mining: Vision, Benefits, and Challenges. IEEE Access **9**, 99480–99494 (2021)
72. Tour, A., Polyvyanyy, A., Kalenkova, A., Senderovich, A.: Agent Miner: An Algorithm for Discovering Agent Systems from Event Data. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds.) Business Process Management. pp. 284–302. Lecture Notes in Computer Science, Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-41620-0_17
73. Van Glabbeek, R.J., Weijland, W.P.: Branching time and abstraction in bisimulation semantics. Journal of the ACM **43**(3), 555–600 (May 1996). https://doi.org/10.1145/233551.233556, https://dl.acm.org/doi/10.1145/233551.233556
74. van der Werf, J.M.E.M., Polyvyanyy, A., van Wensveen, B.R., Brinkhuis et al., M.: All that glitters is not gold: Four maturity stages of process discovery algorithms. Inf. Syst. **114**, 102155 (Mar 2023)
75. 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)
76. 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)
77. 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)