Computers in Industry 164 (2025) 104170

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Operational process monitoring: An object-centric approach

Gyunam Park *, Wil M.P. van der Aalst

Process and Data Science Group (PADS), RWTH Aachen University, Ahornstraße 55, Aachen, 52074, North Rhine-Westphalia, Germany

ARTICLE INFO

Keywords:
Operational process monitoring
Object-centric process mining
Decision support systems
Operational support

ABSTRACT

In business processes, an operational problem refers to a deviation and an inefficiency that prohibits an organization from reaching its goals, e.g., a delay in approving a purchase order in a Procure-To-Pay (P2P) process. Operational process monitoring aims to assess the occurrence of such operational problems by analyzing event data that record the execution of business processes. Once the problems are detected, organizations can act upon the corresponding problems with viable actions, e.g., adding more resources, bypassing problematic activities, etc. A plethora of approaches have been proposed to implement operational process monitoring. The lion's share of existing approaches assumes that a single case notion (e.g., a purchase order in a P2P process) exists in a business process and analyzes operational problems defined over the single case notion. However, most real-life business processes manifest the interplay of multiple interrelated objects. For instance, an execution of an omnipresent P2P process involves multiple objects of different types, e.g., purchase orders, goods receipts, invoices, etc. Applying the existing approaches in these object-centric business processes results in inaccurate or misleading results. In this study, we propose a novel approach to assessing operational problems within object-centric business processes. Our approach not only ensures an accurate assessment of existing problems but also facilitates the analysis of object-centric problems that consider the interaction among different objects. We evaluate this approach by applying it to both simulated business processes and real-life business processes.

1. Introduction

Real-life business processes are characterized by the interplay of multiple interconnected objects, making them *object-centric* (van der Aalst, 2019; Galanti et al., 2023). For example, a Purchase-To-Pay (P2P) process involves various object types such as *purchase orders*, *goods receipts*, and *invoices*. An event occurring within these processes might be linked to several objects, each of a unique type. Fig. 1 shows an execution of a P2P process that involves four objects of different types: *purchase order 1*, *goods receipt 1*, *goods receipt 2*, and *invoice 1*. *Verifying goods receipts* is associated with two goods receipts and the corresponding purchase order, whereas *three-way matching* is associated with an invoice as well as the corresponding goods receipts and purchase order to ensure the amount of the invoice.

Process mining aims to analyze business processes from recorded event data. Conventionally, process mining operates on the presumption of a single case notion in an event log, meaning each event is linked to a single case. To build such event logs, it is necessary to establish a *correlation* between events, hinged on a case notion. Fig. 2(a) illustrates the correlated events of the process execution demonstrated in Fig. 1, predicated on the case notion that brings together a purchase order and all related goods receipts (i.e., *purchase order 1*, *goods receipt 1*, and *goods receipt 2*).

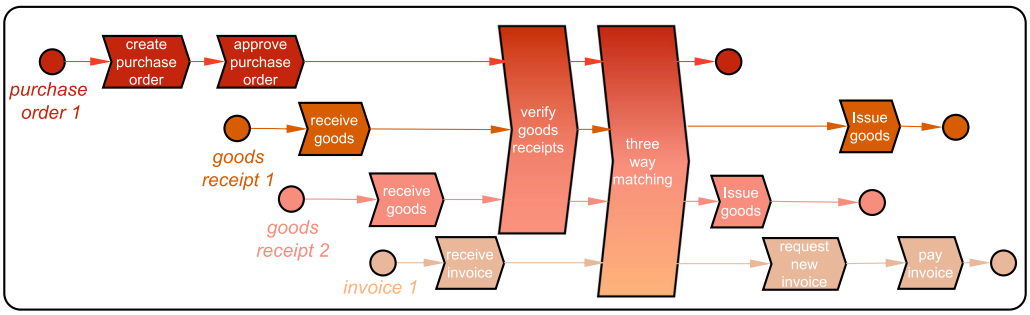
However, such event logs with single case notions suffer from *deficiency* (missing events), *convergence* (duplicated events), and *divergence* (misleading causality between events) issues (van der Aalst, 2020). Fig. 2(a) demonstrates the issues: (1) events related to *invoices*, i.e., *receive invoice*, *request new invoice*, *pay invoice*, are missing (deficiency), (2) *verify goods receipt* is duplicated three times, i.e., once for *purchase order 1*, *goods receipt 1*, and *goods receipt 2* (convergence), (3) the first *receive goods* is causally related to the second *receive goods* (divergence).

In the context of process mining, operational process monitoring refers to the assessment of *operational problems* in business processes by analyzing event logs (dos Santos Garcia et al., 2019). Numerous approaches have been developed to aid this monitoring, each utilizing distinct graphical notations to formally represent these problems. These representations greatly simplify the formal depiction of operational problems, offering a user-friendly alternative to traditional query languages like SQL. This simplification is essential as it enables stakeholders to design and monitor operational problems more efficiently.

However, existing techniques assume an event is associated with a single case notion and analyze event logs containing such events. Therefore, the techniques encounter problems inherent in the event

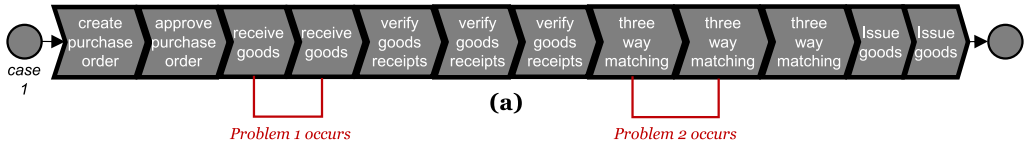
* Corresponding author.
E-mail addresses: gnpark@pads.rwth-aachen.de (G. Park), wvdaalst@pads.rwth-aachen.de (W.M.P. van der Aalst).

https://doi.org/10.1016/j.compind.2024.104170
Received 26 April 2024; Received in revised form 29 August 2024; Accepted 30 August 2024
Available online 10 September 2024



(a) A process execution in a P2P process: reality

Fig. 1. A process execution in a P2P process: reality.



(a)

- *Problem 1*: *Receive goods* is not directly followed by *verify goods receipts*
- *Problem 2*: *Three way matching* is not directly followed by *issue goods*
- *Problem 3*: *Request new invoice* or *pay invoice* both occur
- *Problem 4*: The execution of *receive goods* does not involve the corresponding *purchase order*
- *Problem 5*: The execution of *verify goods receipts* involve more than 5 *goods receipts*
- *Problem 6*: The execution of *three way matching* involves more than 1 *invoice*

(b)

O: the problem occurs, X: the problem does not occur, NA: the process monitoring is not available

	Problem 1	*Problem 2*	*Problem 3*	*Problem 4*	*Problem 5*	*Problem 6*
Reality	X	X	O	O	X	X
Existing approaches	O	O	X	NA	NA	NA

(c)

Fig. 2. (a) A sequence of the events correlated by a single case notion from the process execution shown in Fig. 1. (b) Examples of operational problems. (c) Misleading monitoring results produced by existing approaches.

logs. Fig. 2(b) shows some operational problems defined over the P2P process. Existing techniques determine that *Problem 1* applies in Fig. 2(a) as the *receive goods* is directly followed by the second *receive goods*. However, this is not the case in reality as each *receive goods* event is directly followed by a *verify goods receipt* event for each goods receipt. Furthermore, *Problem 2* is assessed to be applicable since the first *three-way matching* is not directly followed by *issue goods*. However, in reality, each *three-way matching* event is directly followed by an *issue goods* event for each goods receipt. In addition, *Problem 3* is deemed non-applicable as neither *request new invoice* nor *pay invoice* events occur, while in reality, it is applicable.

In this work, we aim to develop an approach to object-centric operational process monitoring that handles the issues caused by the presence of single-case notions in event logs. To that end, we exploit Object-Centric Event Logs (OCELs) (Ghahfarokhi et al., 2021) that preserve the interaction of different objects, as shown in Fig. 1. In addition to correctly evaluating operational problems (e.g., *Problem 1-3*), the proposed object-centric operational process monitoring supports a broader range of object-centric operational problems that involve the interaction of different objects. For example, we can analyze whether the execution of an activity involves unnecessary or missing objects (cf. *Problem 4*). Furthermore, we can examine the cardinality of objects required for executing an activity (cf. *Problem 5* and *Problem 6*).

To this end, we first introduce object-centric operational problems our approach aims to assess. Subsequently, we introduce object-centric behavioral metrics derived from Object-Centric Event Logs (OCELs), which serve to quantify these operational problems. Utilizing these metrics, we design *Object-Centric Problem Graphs (OPGs)*, which formally represent the problems. Finally, we detail our approach to assessing the occurrence of these operational problems by analyzing OCELs.

In the remainder of this paper, we first discuss the related work in Section 2. Next, Section 3 explains an object-centric event log as an input to our approach. Then, in Section 4, we introduce relevant object-centric operational problems that we aim to focus on in this work. Afterward, we present OPGs to formally represent object-centric operational problems and introduce object-centric process monitoring to evaluate the occurrence of the problems in Section 6. In Section 7, we evaluate the proposed approach with use cases based on simulated business processes and a real-life business process. Lastly, Section 8 discusses the implications and limitations, and Section 9 concludes the paper.

2. Related work

This section presents existing operational process monitoring techniques that support the assessment of various operational problems and highlight the limitations of the existing techniques. We then introduce

object-centric process mining, the basis for developing object-centric operational process monitoring methods.

2.1. Operational process monitoring

Various approaches have been proposed to (1) formally represent problems in business processes in specific formats and (2) assess the occurrence of operational problems based on the representation. Such approaches include *conformance checking-based* (Lee et al., 2018; van Zelst et al., 2019; Burattin et al., 2018; Ramezani et al., 2012), *pattern-based* (Türetken et al., 2012; Santos et al., 2011; Awad et al., 2015), *logic-based* (Hallé and Villemaire, 2008; Basin et al., 2015; Bragaglia et al., 2011), *Declare-based* (Pesic et al., 2007; Montali et al., 2013; Maggi et al., 2011), *graph-based* (Ly et al., 2010; Hildebrandt et al., 2011; Knuplesch et al., 2017), *Complex Event Processing (CEP)-based* (Cugola and Margara, 2012; Daum et al., 2012; Weidlich et al., 2011, 2014), and *Instance Spanning Constraints (ISP)-based* (Leitner et al., 2012; Winter and Rinderle-Ma, 2017; Indiono et al., 2016) approaches.

First, conformance-checking techniques are lifted to runtime to allow for the monitoring of non-conforming behaviors. Such techniques evaluate the conformance of running process instances using imperative process models (i.e., process models that describe precise end-to-end control flows). Alignments are often used to check the conformance of process instances (de Leoni and Marrella, 2017). van Zelst et al. (2019) suggest a technique for computing prefix alignments to uncover deviant behavior. Moreover, Burattin et al. (2018) propose a generic framework to derive conformance indicators based on behavioral patterns. Moreover, alignment of Petri-net-based constraints with event logs is performed to assess whether the business process executions adhere to the constraints (Ramezani et al., 2012).

Pattern-based methods establish rules or constraints that describe business processes' compliance requirements either as 'patterns' denoting acceptable behavior or 'anti-patterns' specifying unacceptable behavior. Türetken et al. (2012) suggest a pattern-based framework for capturing and managing business process compliance requirements. Santos et al. (2011) propose task ordering and resource involvement patterns in process instances. Additionally, Awad et al. (2015) establish a set of generic patterns related to task occurrence, their sequencing, and resource allocation, thereby producing anti-patterns from these patterns for event execution monitoring.

Logic-based methods represent compliance requirements as first-order logic and its derivatives. An approach rooted in LTL-FO$^+$, a type of linear temporal logic, is provided by Hallé and Villemaire (2008). LTL-FO$^+$ is mainly employed for analyzing data-aware constraints in processes. Instead of yielding binary answers, Bragaglia et al. (2011) employ fuzzy logic to measure the compliance level of the observed process instance.

Several methods represent operational problems as *Declare* constraints, a language that is declarative and based on a flexible array of constraints, including existence, choice, relation, and negation constraints (Burattin et al., 2016). *MobuconEC* (Montali et al., 2013) and *MobuconLTL* (Maggi et al., 2011) convert these *Declare* constraints into Event Calculus and Linear Temporal Logic (LTL), respectively.

Graph-based techniques leverage the intuitiveness of a graph structure to demonstrate problems visually. Ly et al. (2010) suggest a method for instantiating and verifying compliance rules using Compliance Rule Graphs (CRGs), which formulate compliance rules based on first-order predicate logic. Dynamic Condition Response Graphs (DCR Graphs) are proposed by Hildebrandt et al. (2011) to visually denote compliance rules. An Extended Compliance Rule Graph (eCRG) builds on CRGs to monitor multiple perspectives of business processes, such as time, data, and resource (Knuplesch et al., 2017).

In broad terms, stream data management (Chaudhry et al., 2005; Stefanowski et al., 2014) provides process monitoring techniques. The objective of Complex Event Processing (CEP) in stream data management is real-time identification and response to significant events (Bruns et al., 2019). To this end, it processes real-time events and extracts information from the events as they occur. CEP can support the monitoring of problems in business processes, i.e., operational problems are defined as meaningful events, and their monitoring is considered as the identification of such events. Daum et al. (2012) discuss how CEP can be used for monitoring business processes. Weidlich et al. (2011, 2014) develop a method to abstract process models into behavior profiles and generate event queries from the profile, and track rule-violating event executions using CEP engines.

Moreover, several approaches have been proposed to handle problems that span multiple instances, i.e., Instance Spanning Constraints (ISCs) (Leitner et al., 2012). Winter and Rinderle-Ma (2017) formalize the structure and semantics of ISCs based on Event Calculus. Indiono et al. (2016) propose an approach to monitoring ISCs based on Rete algorithm.

However, existing techniques may lead to misleading results in object-centric environments as they overlook the interactions among different types of objects. Furthermore, operational problems peculiar to object-centric settings are not supported by current techniques.

2.2. Object-centric process mining

Our research aligns with advancements in the field of object-centric process mining (van der Aalst, 2019). Conventional process mining necessitates every event to associate with a singular object type (i.e., a single case notion), and each change of case perspective requires new data extraction and configuration (Diba et al., 2020). This requirement can inadvertently duplicate events (convergence) and lose causal information between events (divergence) (van der Aalst, 2020). Furthermore, the single-object focus can overlook interactions with other object types in the process, leading to potentially inaccurate and misleading analysis. The root cause of these limitations is the oversimplification of process realities involving multiple objects, essentially reducing a 3-dimensional reality to 2-dimensional event log process models.

Object-centric process mining challenges the conventional process mining assumption that each event is linked to a singular case notion or object. Instead, it allows for one event to be associated with multiple objects. This approach (1) facilitates a one-time data configuration, empowering process analysts to select their preferred perspective of objects and events following data upload, (2) captures interactions between objects, fostering the analysis of multiple interacting business processes, (3) employs multi-dimensional data models, providing a more precise representation of business processes (van der Aalst, 2023).

Recently, many approaches have been suggested to support process analysis in object-centric settings. To facilitate such analysis, various data transformation techniques have been proposed to transform data from different sources to object-centric event logs. First, techniques are developed to convert data from databases or Enterprise Resource Planning (ERP) systems into object-centric event logs (Xiong et al., 2022; Berti et al., 2023). Furthermore, Rebmann et al. (2022) proposed a method for converting traditional event logs, which typically focus on single case notions, into object-centric event logs. Additionally, Khayatbashi et al. (2023) introduce a technique for transforming event logs stored in graph databases into object-centric event logs.

In parallel, techniques for process discovery, conformance checking, and process enhancement have been developed for object-centric process mining. First, Li et al. (2017) introduced Object-Centric Behavioral Constraint (OCBC) models, which integrate data models with process models. While the data model illustrates complex interactions between object types, the process model outlines activity order. Berti and van der Aalst (2019) suggest Multiple View Point (MVP) models to address the inherent complexity and scalability issues of OCBC models. Additionally, van der Aalst and Berti (2020) propose a process discovery technique that takes object-centric event logs as the input and produces object-centric Petri nets as the output.

Conformance checking techniques have also been developed. One such technique (van der Aalst and Berti, 2020) employs a token-based replay. This technique "plays a token game" for each object type by flattening the event log to each object type and then merging the results. Adams and van der Aalst (2021) propose a method to determine precision and fitness without the need to flatten object-centric event logs. Recently, Liss et al. (2023) suggest an object-centric alignment technique by using a synchronous product net incorporating a process execution net and a de-jure process model.

On the enhancement front, Esser and Fahland (2021) introduced a query-based approach using graph databases as the storage medium for object-centric event data. This approach allows users to perform complex queries and compute statistics such as temporal relationships and path lengths between events. Berti and van der Aalst (2023) have developed a tool for calculating object-centric performance measures, including the duration times of different objects within the logs. Park et al. (2022) proposed a technique to analyze performance metrics in relation to object-centric Petri nets. This method encompasses the discovery of object-centric Petri nets from event logs, aligning events within these logs to the nets, and computing various performance metrics, such as synchronization time and pooling time.

Object-centric predictive monitoring enhances predictive monitoring techniques by encompassing features that encapsulate the interactions among objects. Galanti et al. (2020) suggested a predictive analytics framework for object-centric event logs, which enhances traditional predictive process analytics by considering object interactions as additional features. Adams et al. (2022) proposed an approach that represents process instances as event graphs of interacting objects, thereby preserving the inherent graph structures found in object-centric event logs. Moreover, a graph embedding approach (Adams et al., 2023) is proposed to effectively embed the structural integrity of process instances in vector format for sequential machine learning models.

Our proposed method for object-centric operational process monitoring substantially builds upon and extends the existing advancements in object-centric process mining. Park and van der Aalst (2022) present initial endeavors to monitor operational problems in object-centric processes, with a focus on readily available simple operational problems. This work extensively extends the existing work by suggesting a comprehensive list of operational problems and developing graphical notations to support the monitoring of operational problems in multi-dimensional aspects.

3. Background: Object-centric event logs

This section introduces an object-centric event log as the input to our proposed approach. An object-centric event log overcomes the limitation of traditional event logs by relating events to multiple interacting objects (van der Aalst, 2019). The definition of object-centric event logs requires the preliminary introduction of several universes.

Definition 1 (*Universes*). $\mathbb{U}_{oi}$ denotes the set of all possible object identifiers, $\mathbb{U}_{ot}$ denotes the set of all possible object types, $\mathbb{U}_{ei}$ denotes the set of all possible event identifiers, $\mathbb{U}_{act}$ denotes the set of all possible activity names, $\mathbb{U}_{time}$ denotes the set of all possible timestamps, $\mathbb{U}_{attr}$ denotes the set of all possible attributes, $\mathbb{U}_{val}$ denotes the set of all possible values, and $\mathbb{U}_{map} = \mathbb{U}_{attr} \nrightarrow \mathbb{U}_{val}$ denotes the set of all possible attribute-value mappings. Each universe consists of unique and non-overlapping sets, e.g., $\mathbb{U}_{oi} \cap \mathbb{U}_{ei} = \emptyset$. Given $map \in \mathbb{U}_{map}$ and $d \notin dom(map)$, $map(d) = \perp$.

Upon establishing these universes, an object-centric event log is defined in the following.

Definition 2 (*Object-Centric Event Log (OCEL)*). An object-centric event log, denoted by EL, is a quadruple $EL = (O, E, R, \mu)$, where

Table 1
A fragment of an event log.

Event id	Activity	Timestamp	Order	Item
e_{9301}	*place order (po)*	25-05-2023:09.35	$\{o_{2201}\}$	$\{i_{8301}, i_{8302}, i_{8303}\}$
e_{9302}	*evaluate credit (ec)*	25-05-2023:13.35	$\{o_{2201}\}$	$\emptyset$
e_{9303}	*confirm order (co)*	25-05-2023:15.35	$\{o_{2201}\}$	$\{i_{8301}, i_{8302}, i_{8303}\}$
e_{9304}	*place order (po)*	26-05-2023:11.20	$\{o_{2202}\}$	$\{i_{8304}, i_{8305}\}$
e_{9305}	*cancel order (ca)*	26-05-2023:11.50	$\{o_{2202}\}$	$\{i_{8304}, i_{8305}\}$

- $O \subseteq \mathbb{U}_{oi}$ represents a collection of objects;
- $E \subseteq \mathbb{U}_{ei}$ represents a collection of events;
- $R \subseteq E \times O$ represents relationships between the events and objects, respectively;
- $\mu \in (E \rightarrow \mathbb{U}_{map}) \cup (O \rightarrow \mathbb{U}_{map})$ represents a mapping function.

For any object, $o \in O$, $\mu(o)(type) \in \mathbb{U}_{ot}$ holds, and for any event $e \in E$, we ensure that $\mu(e)(act) \in \mathbb{U}_{act}$ and $\mu(e)(time) \in \mathbb{U}_{time}$. $\prec_{EL} \subset E \times E$ is a total order such that for any pair of events $e_1, e_2 \in E : e_1 \prec_{EL} e_2$ implies $e_1 \neq e_2$ and $\mu(e_1)(time) \leq \mu(e_2)(time)$. The set of all possible logs is denoted by $\mathbb{U}_{EL}$.

For convenience, we will express $\mu(o)(x)$ as $\mu_x(o)$ and $\mu(e)(x)$ as $\mu_x(e)$. Table 1 depicts a subset of an event log $EL_1 = (O_1, E_1, R_1, \mu_1)$ with $O_1 = \{o_{2201}, o_{2202}, i_{8301}, i_{8302}, \dots\}$, $E_1 = \{e_{9301}, e_{9302}, \dots\}$, $R_1 = \{(o_{2201}, e_{9301}), (i_{8301}, e_{9301}), \dots\}$, $\mu_{act}(e_{9301}) = po$, $\mu_{time}(e_{9301}) = 25\text{–}05\text{–}2023:09.35$, etc.

We introduce functions to facilitate queries from event logs as follows:

Definition 3 (*Log-Related Symbols*). For $EL = (O, E, R, \mu) \in \mathbb{U}_{EL}$, the following symbols are introduced:

- $act(EL) = \{\mu_{act}(e) \mid e \in E\}$, the collection of activities;
- $event(EL, a) = \{e \in E \mid \mu_{act}(e) = a\}$, the collection of events associated with activity $a \in act(EL)$;
- $type(EL) = \{\mu_{type}(o) \mid o \in O\}$, the collection of object types;
- $object(EL, ot) = \{o \in O \mid \mu_{type}(o) = ot\}$, the collection of objects associated with object type $ot \in type(EL)$;
- $event(EL, o) = \{e \in E \mid (e, o) \in R\}$, the collection of events involving object $o \in O$;
- $object(EL, e) = \{o \in O \mid (e, o) \in R\}$, the collection of objects participating in event $e \in E$;
- $object(EL, e, ot) = \{o \in O \mid (e, o) \in R \wedge \mu_{type}(o) = ot\}$, the collection of objects of objec type $ot \in type(EL)$ participating in event $e \in E$;
- $sequence(o) = \langle e_1, e_2, \dots, e_n \rangle$ s.t. $event(EL, o) = \{e_1, e_2, \dots, e_n\}$ and $e_i \prec_{EL} e_j$ for any $1 \leq i < j \leq n$ is the ordered series of all events in which object $o \in O$ participates;
- $\sigma(o) = \langle a_1, a_2, \dots, a_n \rangle$ s.t. $sequence(o) = \langle e_1, e_2, \dots, e_n \rangle$ and $a_i = \mu_{act}(e_i)$ for any $1 \leq i \leq n$ is the trace of object $o \in O$.

For example, $act(EL_1) = \{place\ order\ (po), evaluate\ credit\ (ec), confirm\ order\ (co), cancel\ order\ (co)\}$, $event(EL_1, po) = \{e_{9301}, e_{9304}\}$, $type(EL_1) = \{order, item\}$, $object(EL_1, order) = \{o_{2201}, o_{2202}\}$, $event(EL_1, o_{2201}) = \{e_{9301}, e_{9302}, e_{9303}\}$, $object(EL_1, e_{9301}) = \{o_{2201}, i_{8301}, i_{8302}, i_{8303}\}$, $sequence(o_{2201}) = \langle e_{9301}, e_{9302}, e_{9303} \rangle$, and $trace(o_{2201}) = \langle po, ec, co \rangle$. Note that the definition of sequences relies on a total order $\prec_{EL} \subset E \times E$ that is formally defined in Definition 2. We define that $\prec_{EL}$ is a strict total order, thus avoiding any ambiguity in the sequence of events caused by, e.g., identical timestamps. Moreover, the membership operator $\in$ and the subset operator $\subseteq$ are applied to $\sigma(o)$ as if it were a set. Therefore, $a \in \sigma(o)$ if $a \in event(EL, o)$ and $\{a, b\} \subseteq \sigma(o)$ if $\{a, b\} \subseteq event(EL, o)$.

4. Object-centric operational problems

In this section, we present a list of object-centric operational problems that we aim to analyze using the proposed approach. Operational

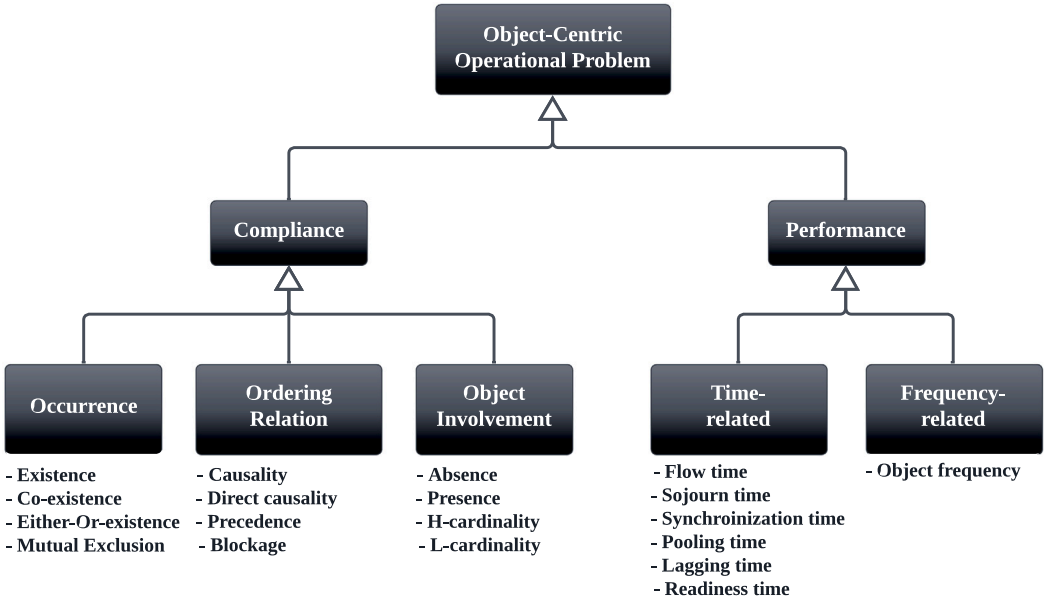


Fig. 3. Taxonomy of object-centric operational problems in UML 2.0 class diagram.

problems are classified into two primary categories: *task-oriented* and *resource-oriented*. The former category addresses issues directly related to the execution of tasks within operational processes, e.g., the compliance and performance of the task executions. The latter category focuses on the management and efficiency of resources required to execute these tasks, e.g., the authorization of resources and separation of duties.

Focusing on the task-oriented problems, we classify object-centric operational problems into *compliance* and *performance* problems as shown in Fig. 3. The former concerns the violation of compliance rules, whereas the latter concerns bad performances that are measured by various performance metrics.

Next, we further distinguish three types of compliance problems: *occurrence*, *ordering relation*, and *object involvement*. Moreover, we classify performance problems into *time-related problems* and *frequency-related problems*.

In the following, we explain object-centric operational problems in each category in more detail, using the running example of an Order-To-Cash (O2C) process described in Fig. 4. The process is described as an object-centric Petri net (van der Aalst and Berti, 2020). An object-centric Petri net comprises places, transitions, arcs, and variable arcs as visualized in the legend of Fig. 4. Each place is designated by a specific color that represents different object types involved in the process. A transition is *fired* by consuming tokens from all the incoming places and producing tokens to all the outgoing places. A single token is consumed/produced if the incoming/outgoing place is connected via arcs, while a variable number of tokens can be consumed/produced if the incoming/outgoing place is connected via variable arcs.

4.1. Object-centric compliance problems

An object-centric compliance problem concerns compliance rules that organizations are ascertained to adhere to. Such rules originate from various sources, such as the laws and regulations that legislation and regulatory bodies have enforced, standards (e.g., ISO 9000), and internal policies. Service Level Agreements (SLAs) also often contribute to compliance problems, particularly when they specify conditions that align with regulatory or standard compliance. Compliance problems are further categorized into *occurrence*, *ordering relation*, and *object involvement* problems. In the following section, we explain each category and its relevant problems.

First, occurrence-related compliance problems refer to operational problems arising from the presence or absence of certain activities within a process instance. These problematic situations include:

- *Existence* points to an issue arising from the presence of activity within an object's lifecycle. An example of this is when the activity *send reminder via email* exists in over 80% of all *orders*.
- *Co-existence* highlights a problem where two activities both occur within an object's lifecycle. An example of this is when both activities *send invoice via email* and *send reminder via post* co-exist in more than 5% of all *orders*.
- *Either-Or-existence* denotes an issue related to the presence of at least one of two activities within an object's lifecycle. An example of this is when, in over 90% of all items, either *skip availability checking* or *cancel item* occurs.
- *Mutual exclusion* refers to a problematic situation where two activities mutually exclude each other within an object's lifecycle. An example of this is when, for more than 5% of all items, the activities *check availability* and *pick item* never co-occur.

Second, ordering relation-related compliance problems refer to operational problems related to the ordering relation of activities in process instances. It includes the following problems:

- *Causality* pertains to the causal relationship between two activities within the lifecycle of an object. An example of this is when the activity *collect payment* is followed by *send invoice via email* in more than 1% of all *orders*.
- *Direct causality* deals with the direct causal relationship between two activities within an object's lifecycle. An example of this is when the activity *send invoice via post* is directly followed by *send reminder via post* in less than 10% of all *orders*.
- *Precedence* concerns the precedence relation of two activities in the lifecycle of an object. An example of this is when *pick item* precedes *check availability* in more than 5% of all *items*.
- *Blockage* relates to the blocking relationship between two activities within an object's lifecycle. An example of this is when *send invoice via post* always prevents *send invoice via email* from happening.

Third, object involvement-related compliance problems refer to issues associated with how objects partake in the execution of an activity. It includes the following problems:

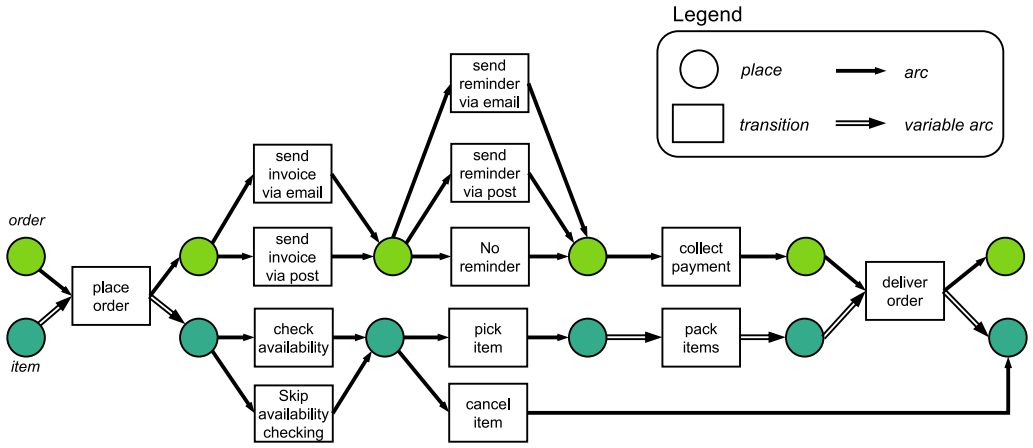


Fig. 4. An O2C process as a running example in an object-centric Petri net. The process consists of two object types: *order* and *item*. First, an order is placed with a variable number of items. On the one hand, for each order, the corresponding invoice is sent either via post or email. A reminder is sent via post or email, if necessary, and the corresponding payment is collected. On the other hand, for each item, the availability may be checked. An item is picked if available and canceled otherwise. The picked items are packed, and finally, the items of an order are delivered.

- *Absence* refers to a situation where the absence of an object type in an activity execution is problematic. An example of this is when the execution of *deliver order* without any *items* occurs in more than 1% of cases.
- *Presence* refers to a situation where the presence of an object type in the execution of an activity is problematic. An example of this is when the execution of *deliver order* with unknown *quotation documents* occurs in more than 1% of cases.
- *Cardinality* refers to issues arising from the number of object participations in carrying out an activity, either being too low or too high. This problem encompasses:
  - *Higher Cardinality:* Concerns occur when the participation of an object type exceeds a certain threshold. An example of this is when the activity *collect payment* includes multiple unrelated *orders* in more than 5% of cases, which could complicate payment processing or increase the risk of errors.
  - *Lower Cardinality:* Issues arise when the participation of an object type is less than a desired minimum. For example, a situation where the activity *place order* typically involves only one *item* in more than 30% of cases may suggest an issue with order batching or promotional strategy.

4.2. Object-centric performance problems

An object-centric performance problem relates to Key Performance Indicators (KPIs). Based on the associated KPIs, we classify these problems into two types: *time-related performance* problems and *frequency-related performance* problems.

Firstly, time-related performance problems pertain to time-specific KPIs established over the execution of an activity. For example, the average time taken to *deliver order* might be a concern if it consistently exceeds two days. Further, some time-related performance problems relate to KPIs that are defined concerning a particular object type, such as the average pooling/lagging/queuing times of *deliver order* concerning *items* exceeding 12 h.

Secondly, frequency-related performance problems are concerned with KPIs related to the frequency of an object type in the execution of an activity. A problem like "The average number of *items* in the execution of *place order* is higher than 10" is a frequency-related performance problem defined over the frequency of *items* in the execution of *place order*.

5. Object-centric behavioral metrics

In this section, we introduce the concept of object-centric behavioral metrics. An object-centric behavioral metric quantifies compliance and performance issues in object-centric business processes. In this section, we introduce object-centric behavioral metrics that quantify object-centric problems introduced in Section 4. For example, *co-exist* (cf. Definition 4) quantifies the co-existence problem, e.g., the co-existence of *send reminder via email* and *send reminder via post* within the lifecycle of an *order* by calculating the ratio of *orders* for which the *send reminder via email* event and *send reminder via post* event co-exist.

5.1. Object-centric occurrence

An object-centric occurrence metric concerns the occurrence of an activity or multiple activities w.r.t. the lifecycles of an object.

Definition 4 (*Occurrence Metrics*)**.** Let $EL = (O, E, R, \mu) \in \mathbb{U}_{EL}$ be an event log, $a, b \in act(EL)$ activities, and $ot \in type(EL)$ an object type.

- $exist(EL, ot, a) = \frac{|\{o \in object(EL,ot) | a \in \sigma(o)\}|}{|object(EL,ot)|}$ is the ratio of ot objects whose trace contains a;
- $co\text{-}exist(EL, ot, a, b) = \frac{\left|\{o \in object(EL,ot) | \{a,b\} \subseteq \sigma(o) \vee \{a,b\} \cap \sigma(o) = \emptyset\}\right|}{|object(EL,ot)|}$ is the ratio of ot objects whose trace contain both a and b or neither of them;
- $either\text{-}or\text{-}exist(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) | a \in \sigma(o) \vee b \in \sigma(o)\}|}{|object(EL,ot)|}$ is the ratio of ot objects whose trace contain either a or b or both;
- $mutual\text{-}exclusive(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) | \neg(a \in \sigma(o) \wedge b \in \sigma(o))\}|}{|object(EL,ot)|}$ is the ratio of ot objects whose trace does not contain both a and b.

For instance, $exist(EL_1, order, po) = 1$, i.e., *place order* exist for all *orders*. $co\text{-}exist(EL_1, order, po, ca) = 0.5$, i.e., half of the *orders* involve both *place order* and *cancel order*. $either\text{-}or\text{-}exist(EL_1, order, ec, co) = 0.5$, i.e., half of the *orders* involve either *evaluate credit* or *confirm order*. $mutual\text{-}exclusive(EL_1, order, ca, ec) = 1$, i.e., *cancel order* and *evaluate credit* do not co-occur for all *orders*.

5.2. Object-centric ordering relation

An object-centric ordering relation metric concerns the ordering relation of two activities w.r.t. the lifecycles of an object.

Definition 5 (*Ordering Relation Metrics*). Let $EL = (O, E, R, \mu) \in \mathbb{U}_{EL}$ be an event log, $a, b \in act(EL)$ activities, and $ot \in type(EL)$ an object type.

- $cause(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) \mid a \notin \sigma(o) \vee \exists_{1 \le i < j \le |\sigma(o)|} \sigma(o)(i) = a \wedge \sigma(o)(j) = b\}|}{|object(EL,ot)|}$ is the ratio of ot objects where b occurs if a occurs;
- $directly\text{-}cause(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) \mid a \notin \sigma(o) \vee \exists_{1 \le i < |\sigma(o)|} \sigma(o)(i) = a \wedge \sigma(o)(i+1) = b\}|}{|object(EL,ot)|}$ is the ratio of ot objects where b occurs directly after a occurs;
- $precede(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) \mid b \notin \sigma(o) \vee \exists_{1 \le i < j \le |\sigma(o)|} \sigma(o)(i) = a \wedge \sigma(o)(j) = b\}|}{|object(EL,ot)|}$ is the ratio of ot objects where a precedes if b occurs;
- $block(EL, ot, a, b) = \frac{|\{o \in object(EL,ot) \mid a \notin \sigma(o) \vee \exists_{1 \le i \le |\sigma(o)|} \sigma(o)(i) = a \Rightarrow \nexists_{i < j \le |\sigma(o)|} \sigma(o)(j) = b\}|}{|object(EL,ot)|}$ is the ratio of ot objects where b does not occur if a occurs.

For instance, $cause(EL_1, order, po, co) = 0.5$, i.e., for half of the orders, *confirm order* occurs if *place order* occurs. $directly\text{-}cause(EL_1, order, po, co) = 0$, i.e., *confirm order* never occurs directly after *place order* occurs. $precede(EL_1, order, po, ca) = 1$, i.e., *place order* always precedes if *cancel order* occurs. $block(EL_1, order, ca, co) = 1$, i.e., *confirm order* never occurs if *cancel order* occurs.

5.3. Object involvement metrics

An object involvement metric represents the involvement of different objects in the execution of an activity.

Definition 6 (*Object Involvement Metrics*). Let $EL = (O, E, R, \mu) \in \mathbb{U}_{EL}$ be an event log, $ot \in type(EL)$ an object type, and $act \in act(EL)$ an activity.

- $absent(EL, ot, act) = \frac{|\{e \in event(EL,act) \mid |object(EL,e,ot)| = 0\}|}{|event(EL,act)|}$ is the ratio of act events where ot objects are not involved;
- $present(EL, ot, act) = \frac{|\{e \in event(EL,act) \mid |object(EL,e,ot)| > 0\}|}{|event(EL,act)|}$ is the ratio of act events where any number of ot objects are involved;
- $h\text{-}cardinality_k(EL, ot, act) = \frac{|\{e \in event(EL,act) \mid |object(EL,e,ot)| > k\}|}{|event(EL,act)|}$ is the ratio of act events where more than k objects of type ot are involved;
- $l\text{-}cardinality_k(EL, ot, act) = \frac{|\{e \in event(EL,act) \mid |object(EL,e,ot)| < k\}|}{|event(EL,act)|}$ is the ratio of act events where less than k objects of type ot are involved.

For instance, $absent(EL_1, item, ec) = 1$, i.e., *evaluate credit* events never involve items. $present(EL_1, order, ec) = 1$, i.e., *evaluate credit* events always involve orders. $h\text{-}cardinality_1(EL_1, item, po) = 1$, i.e., *place order* events always involve multiple items. $l\text{-}cardinality_2(EL_1, order, ec) = 1$, i.e., *evaluate credit* events always involve one order.

5.4. Object-centric performance

An object-centric performance metric is a value indicating the frequency or performance of an activity's execution. To quantify these performance measures for an activity, it is necessary to identify the *enabling* events that have a causal relationship with the activity's execution. For instance, performance metrics of *Deliver Order (DO)* in Fig. 5, i.e., $e10$, is defined with the three enabling events connected by the incoming arcs (i.e., $e7$ and $e9$ labeled *PAck Item (PAI)* and $e8$ labeled Collect Payment (*CP*)). Specifically, a sojourn time of the event is defined as the time difference between $e10$'s timestamp and $e9$'s timestamp.

A relation function relates an event to its enabling events.

Definition 7 (*Relation*). Let $EL = (O, E, R, \mu)$ be an event log. A relation function $rel_{EL} \in E \to \mathcal{P}(E)$ maps an event to a set of its enabling events such that, for any $e \in E$, $rel_{EL}(e) = \{e' \in E \mid \exists_{o \in O \wedge 2 \le i \le |sequence(o)|} sequence(o)(i) = e \wedge sequence(o)(i-1) = e'\}$.

Given EL' described in Fig. 5, $rel_{EL'}(e10) = \{e7, e8, e9\}$, i.e., $e10$ is enabled by $e7$, $e8$, and $e9$.

Definition 8 (*Performance Metrics*). Let $EL = (O, E, R, \mu) \in \mathbb{U}_{EL}$ be an event log, $ot \in type(EL)$ an object type, and $act \in act(EL)$ an activity. Let $\mathbb{U}_{agg} = \mathcal{B}(\mathbb{R}) \to \mathbb{R}$ be the universe of aggregation functions. Let $agg \in \mathbb{U}_{agg}$ be an aggregation function, e.g., $agg(b) = \sum_{x \in b} x$ is sum, $agg(b) = \min_{x \in b} x$ is minimum, and $agg(b) = max_{x \in b} x$ is maximum, for any $b \in \mathcal{B}(\mathbb{R})$.

- $freq_{agg}(EL, ot, act) = agg([object(EL, e, ot) \mid e \in event(EL, act)])$ is the object frequency of ot in the execution of act;
- $flow_{agg}(EL, act) = agg([\mu(e)(time) - min(T_e) \mid e \in event(EL, act)])$ with $T_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e)\}$ is the flow time of act;
- $sojourn_{agg}(EL, act) = agg([\mu(e)(time) - max(T_e) \mid e \in event(EL, act)])$ with $T_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e)\}$ is the sojourn time of act;
- $sync_{agg}(EL, act) = agg([max(T_e) - min(T_e) \mid e \in event(EL, act)])$ with $T_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e)\}$ is the synchronization time of act;
- $pool_{agg}(EL, ot, act) = agg([max(T'_e) - min(T'_e) \mid e \in event(EL, act)])$ with $T'_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e) \wedge \exists_{o \in object(EL,e')} \mu_{type}(o) = ot\}$ is the pooling time of act w.r.t. ot;
- $lag_{agg}(EL, ot, act) = agg([max(T'_e) - min(T_e) \mid e \in event(EL, act)])$ with $T_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e)\}$ and $T'_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e) \wedge \exists_{o \in object(EL,e')} \mu_{type}(o) = ot\}$ is the lagging time of act w.r.t. ot;
- $ready_{agg}(EL, ot, act) = agg([max(T_e) - max(T'_e) \mid e \in event(EL, act)])$ with $T_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e)\}$ and $T'_e = \{\mu(e')(time) \mid e' \in rel_{EL}(e) \wedge \exists_{o \in object(EL,e')} \mu_{type}(o) = ot\}$ is the readiness time of a w.r.t. ot.

Fig. 5 shows examples of the performance metrics for *Delivery Order (DO)*.

6. Analyzing object-centric operational problems

This section introduces our approach for assessing object-centric operational problems, building upon the operational problems discussed in Section 4 and the object-centric behavioral metrics explained in Section 5. First, we introduce the syntax and semantics of an Operational Problem Graphs (OPG) to formally represent an operational problem (or a combination of operational problems). Second, we explain how we analyze OCELs to assess the operational problems represented in OPGs.

6.1. Syntax of Operational Problem Graphs (OPGs)

An Operational Problem Graph (OPG) is a hypergraph (E, V) that consists of nodes V and hyperedges E that connect any number of nodes. An OPG comprises two distinct node types and three kinds of hyperedges. Each edge is linked with an operational problem along with a specification outlining how the evaluation of the related problem should be conducted.

Definition 9 (*Operational Problem Graphs*). Let $OP = \{<, \le, >, \ge, =\}$ be a set of relational operators. An OPG is a hypergraph $opg = (A, OT, E_{OA}, E_{AA}, E_{AOA}, metric_{OA}, metric_{AA}, metric_{AOA}, op, thre)$ where

- $A \subseteq \mathbb{U}_{act}$ is a collection of activity nodes;
- $OT \subseteq \mathbb{U}_{ot}$ is a collection of object type nodes;
- $E_{OA} \subseteq OT \times A$ is a collection of Object type to Activity (OA) edges;
- $E_{AA} \subseteq \{(a, a) \mid a \in A\}$ is a collection of Activity to Activity (AA) edges;
- $E_{AOA} \subseteq A \times OT \times A$ is a collection Activity to Object type to Activity (AOA) edges;

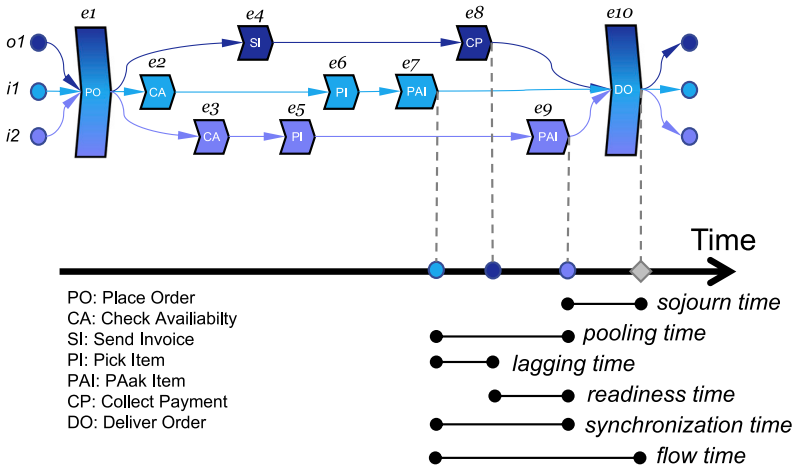


Fig. 5. Top: Event log $EL' = (E', O', \mu', R') \in \mathbb{U}_{EL}$ is graphically represented: chevrons represent events, i.e., $E' = \{e_1, e_2, \dots\}$, circles represent objects, i.e., $O' = \{o_{2201}, i_{8301}, i_{8302}\}$, and arcs represent relation between events and objects. Bottom: performance metrics of *Deliver Order (DO)* are depicted. We relate *DO* event, i.e., $e10$, to its *enabling* events, i.e., $e7$, $e8$, and $e9$, and compute various performance metrics.

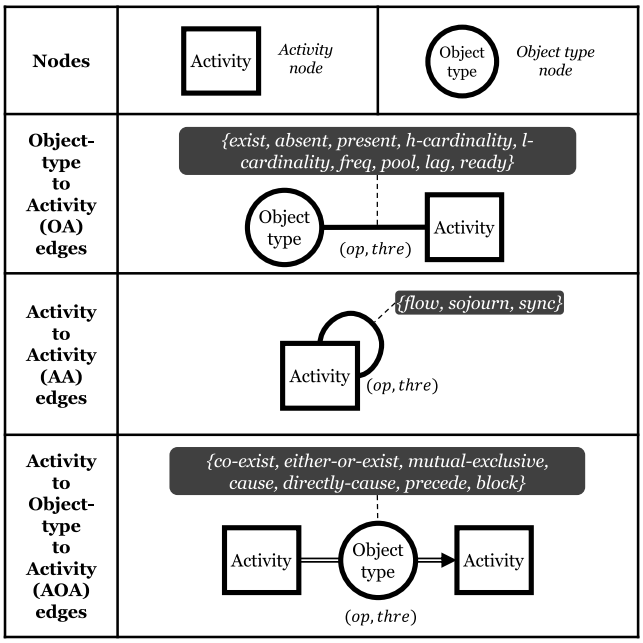


Fig. 6. Graphical notation of OPGs.

- $metric_{OA} \in E_{OA} \to \{exist, absent, present, h\text{-}cardinality_k, l\text{-}cardinality_k\} \cup \{p_{agg} \mid p \in \{freq, pool, lag, ready\} \wedge agg \in \mathbb{U}_{agg}\}$ maps OA edges to behavior metrics
- $metric_{AA} \in E_{AA} \to \{p_{agg} \mid p \in \{flow, sojourn, sync\} \wedge agg \in \mathbb{U}_{agg}\}$ maps AA edges to behavior metrics,
- $metric_{AOA} \in E_{AOA} \to \{co\text{-}exist, either\text{-}or\text{-}exist, mutual\text{-}exclusive, cause, directly\text{-}cause, precede, block\}$ maps AOA edges to behavior metrics,
- $op \in (E_{OA} \cup E_{AA} \cup E_{AOA}) \to OP$ maps edges to relational operators, and
- $thre \in (E_{OA} \cup E_{AA} \cup E_{AOA}) \to \mathbb{R}$ maps edges to thresholds.

The set of all possible OPGs is denoted by $\mathbb{U}_{opg}$.

Fig. 6 shows the graphical notation of OPGs. It consists of two types of nodes (i.e., *activities* as rectangles and *object types* as circles) and three types of edges (i.e., *OA*, *AA*, and *AOA* edges). Each type of edge is associated with a different behavioral metric. First, OA edges are associated with *exist*, *absent*, *present*, *h-cardinality*, *l-cardinality*, *freq*, *pool*, *lag*, and *ready*. Second, AA edges are associated with *flow*, *sojourn*, and *sync*. Finally, AOA edges are associated with *co-exist*, *either-or-exist*, *mutual-exclusive*, *cause*, *directly-cause*, *precede*, and *block*.

Fig. 7 shows several examples of OPGs. For instance, Fig. 7(a) corresponds to $opg_1 = (A_1, OT_1, E_{OA,1}, \emptyset, \emptyset, metric_{OA,1}, \emptyset, \emptyset, op_1, thre_1)$ where $A_1 = \{SRvE\}$, $OT_1 = \{order\}$, $E_{OA,1} = \{oa\text{-}edge_1 = (order, SRvE)\}$, $metric_{OA,1}(oa\text{-}edge_1) = exist$, $op_1(oa\text{-}edge_1) = >$, and $thre_1(oa\text{-}edge_1) = 0.8$.

Note that a composite problem can be represented as OPGs with multiple edges. Graphically, these multiple edges are visualized by curving the edges and labeling them to ensure clarity and to avoid visual clutter. Curving the edges helps distinguish multiple connections edges of the same type, preventing them from overlapping. Edge labeling is crucial as it provides information about each edge.

Fig. 8 shows a composite problem, $opg' \in \mathbb{U}_{opg}$, represented by an OPG with three edges. The problem indicates a problematic situation when *send reminder via post* is followed by *send reminder via email* for more than 80% of orders, while *send reminder via post* is delayed both in terms of average flow times and sojourn times.

6.2. Semantics of OPGs

We define the semantics of OPGs in relation to OCELs using the notion of *occurrence*. An OPG *occurs* in an OCEL if all the conditions represented by different edges of the OPG are satisfied in the OCEL. For example, the OPG depicted in Fig. 7(a) *occurs* in an event log $EL \in \mathbb{U}_{EL}$ if the corresponding behavioral metric, i.e., $exist(EL, order, SRvE)$, satisfies the condition, i.e., the metric is higher than 0.8 in the log.

Definition 10 (*Semantics of OPGs*). Let $EL \in \mathbb{U}_{EL}$ be an event log. Let $opg = (A, OT, E_{OA}, E_{AA}, E_{AOA}, metric_{OA}, metric_{AA}, metric_{AOA}, op, thre)$ be an OPG. Given $EL \in \mathbb{U}_{EL}$, $occur(EL, opg) = true$ if

- for any $oa\text{-}edge = (ot, a) \in E_{OA}$ s.t. $ot \in type(EL)$ and $a \in act(EL)$, $metric_{OA}(oa\text{-}edge)(EL, ot, a) \bigoplus thre(oa\text{-}edge)$, where $\bigoplus = op(oa\text{-}edge)$,
- for any $aa\text{-}edge = (a, a) \in E_{AA}$ s.t. $a \in act(EL)$, $metric_{AA}(aa\text{-}edge)(EL, a) \bigoplus thre(aa\text{-}edge)$, where $\bigoplus = op(aa\text{-}edge)$, and
- for any $aoa\text{-}edge = (a, ot, b) \in E_{AOA}$ s.t. $ot \in type(EL)$, $a, b \in act(EL)$, $metric_{AOA}(aoa\text{-}edge)(EL, ot, a, b) \bigoplus thre(aoa\text{-}edge)$, where $\bigoplus = op(aoa\text{-}edge)$.

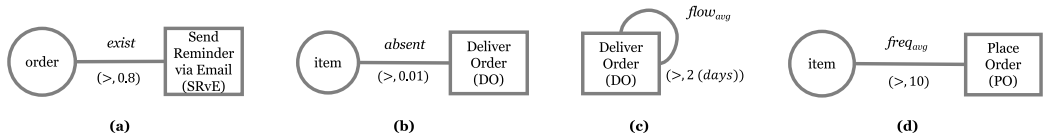


Fig. 7. Examples of OPGs.

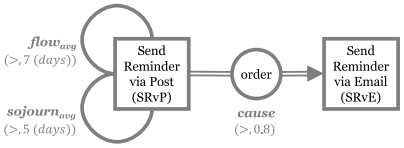


Fig. 8. An OPG representing a composite problem.

Otherwise, $occur(EL, opg) = false$.

For the OPG depicted in Fig. 7(a), opg_1, and $EL' \in \mathbb{U}_{EL}$, $occur(EL', opg_1) = true$ if $exist(EL', order, SRvE) > 0.8$, where $exist = metric_{OA,1}(oa\text{-}edge_1)$ is a metric, $> \; = \; op_1(aa\text{-}edge_1)$ is a relational operator, and $0.8 = thre_1(aa\text{-}edge_1)$ is a threshold.

6.3. Object-centric process monitoring

For a given object-centric event log and an OPG, an object-centric process monitoring evaluates if the OPG *occurs* in an object-centric event log using the semantics of the OPG.

Definition 11 (*Object-Centric Process Monitoring*)**.** An object-centric process monitoring, denoted as $mon \in \mathbb{U}_{EL} \times \mathbb{U}_{opg} \rightarrow \{true, false\}$, is a function that associates an object-centric event log and an OPG with a Boolean value. For any given $EL \in \mathbb{U}_{EL}$ and $opg \in \mathbb{U}_{opg}$, $mon(EL, opg) = true$ if $occur(EL, opg) = true$. Otherwise, it results in $false$.

7. Evaluation

The proposed approach is fully implemented as a web application. This application facilitates the design, storage, and assessment of Operational Problem Graphs (OPGs), and it also enables the visualization of assessment results. The implementation includes three main sub-components: the *OPG Editor*, *OPG Repository*, and *Assessment and Visualization* module. Detailed documentation for the application is available at https://proact.readthedocs.io/en/latest/monitoring_engine/, and the source code can be accessed publicly at https://github.com/gyunamister/proact. We utilize this web application to conduct the following evaluation.

The evaluation consists of two parts. First, we implement simulated use cases focusing on a manufacturing process and a Procure-To-Pay (P2P) process, applicable to a hypothetical company operating on an SAP ERP system. This helps evaluate the feasibility of the approach proposed. Next, we show a real-life use case to evaluate the effectiveness of the object-centric process monitoring and analyze its sensitivity on the threshold of operational problems and the size of event logs.

7.1. Use cases using simulations

We initially present two use cases centered on the manufacturing process and the P2P process.

7.1.1. Use case on manufacturing process

This process incorporates four types of objects: *reservation*, *production order*, *purchase order*, and *purchase requisition*. Fig. 9 shows a process model of the manufacturing process, which involves the four object types, using Object-Centric Petri Nets (OCPNs) as a modeling formalism (van der Aalst and Berti, 2020).

We simulate the process using discrete-event simulation (van der Aalst, 2015) and CPN Tools (Jensen et al., 2007), using parameters such as production lead times, approval thresholds, and material requisition frequencies. The simulation was conducted over three distinct monthly time windows, i.e., Jan. 2022, Feb. 2022, and Mar. 2022.

Next, we extract three OCELs that contain events of different time windows; EL_{man}^{Jan22}, EL_{man}^{Feb22}, and EL_{man}^{Mar22} are OCELs from Jan. 2022, Feb. 2022, and Mar. 2022.[1] Operational challenges that could be encountered within the process include:

- *Bypassing PRA (Purchase Requisition Approval)*: A purchase requisition may occasionally omit the approval stage.
- *Absent reservation for PRA (Purchase Requisition Approval)*: The event of *approving a purchase requisition* sometimes fails to include the corresponding reservation.
- *Excessive reservations per PO (Production Order)*: The event of *creating a production order* tends to involve an above-average number of reservations.
- *Delayed POR (Purchase Order Release)*: *Releasing a purchase order*'s average sojourn time reaches or exceeds 15 days.

We design operational problems that correspond to the operational problems and assess their occurrence in each event log. Fig. 10(a–d) represent *Bypassing PRA*, *Absent reservation for PRA*, *Excessive reservations per PO*, and *Delayed POR*, respectively. Table 2 shows the analysis result. For all the operational problems in all the different time windows, the process monitoring successfully detects the occurrence.

7.1.2. Use case on P2P process

We then proceed to a use case based on the P2P process. This process features five object types: *material*, *purchase requisition*, *purchase order*, *invoice*, and *goods receipt*. Fig. 11 exhibits a model of the process.

We simulate the process and extract three OCELs,[2] each encompassing events from distinct time windows: EL_{p2p}^{1} begins on 01.08.2021 and ends on 14.10.2021, EL_{p2p}^{2} starts on 15.10.2021 and finishes on 18.01.2022, and EL_{p2p}^{3} begins on 01.02.2022 and finishes on 16.05.2022. Potential operational issues in the process could include:

- *Clearance of MIs (Multiple Invoices)*: The event of *clearing an invoice* frequently involves several invoices.
- *Excessive materials per PO (Purchase Order)*: The event of *creating a purchase order* regularly involves a minimum of five materials.
- *Delayed POC (Purchase Order Creation)*: *Creating a purchase order*'s average sojourn time exceeds three days.

We design operational problems corresponding to the operational problems and monitor their occurrence in each event log. Fig. 12(a-c) represent *Clearance of MIs*, *Excessive materials per PO*, and *Delayed POC*, respectively. As shown in Table 3, the process monitoring successfully detects the operational problems in all the different time windows.

[1] The dataset is publicly available at https://github.com/gyunamister/proact/tree/master/samples/logs/production/

[2] The dataset is publicly available at https://github.com/gyunamister/proact/tree/master/samples/logs/p2p/

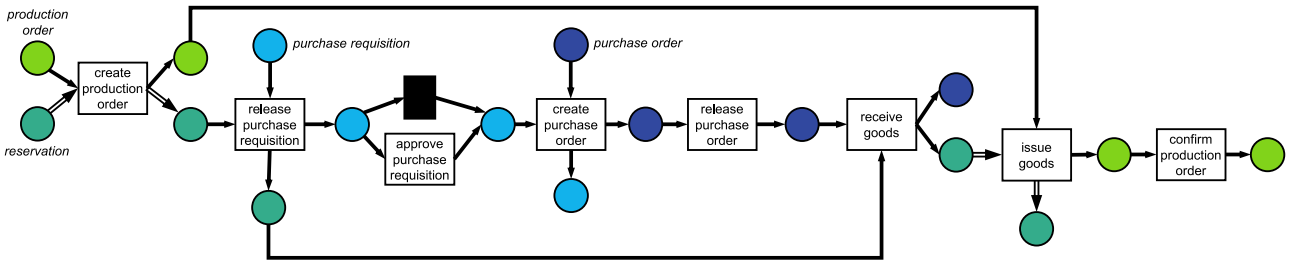


Fig. 9. During the manufacturing process, a production order with a varying number of reservations (denoting required materials) is created initially. Following this, a purchase requisition is released and approved, leading to the creation of a purchase order based on the initial requisition. Once the purchase order is released, the reservations are acknowledged and prepared for production. The process concludes with the confirmation of the production order.

Fig. 10. OPGs representing the operational problems of the manufacturing process.

Table 2
Analysis results of the manufacturing process. ✓ and ✗ denote the occurrence and non-occurrence of the problem, respectively.

Problems	Event Log					
	EL_{prod}^{Jan22}		EL_{prod}^{Feb22}		EL_{prod}^{Mar22}	
	Reality	*Monitoring*	*Reality*	*Monitoring*	*Reality*	*Monitoring*
Bypassing PRA	✗	✗	✓	✓	✓	✓
Absent reservation for PRA	✓	✓	✓	✓	✓	✓
Excessive reservation per PO	✓	✓	✓	✓	✓	✓
Delayed POR	✗	✗	✓	✓	✗	✗

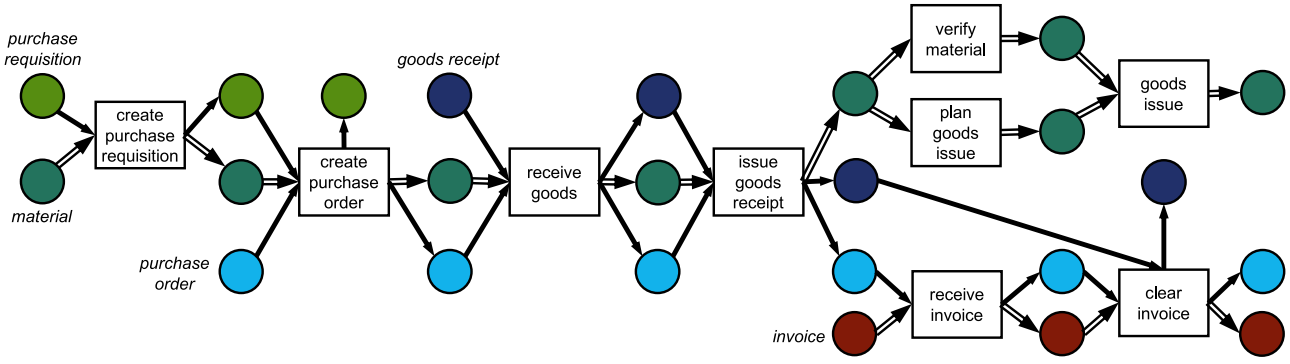


Fig. 11. In the P2P process, the initial step involves creating a purchase requisition accompanied by several materials. Subsequently, a purchase order is created with the requisition and materials. The goods are then received and issued to different organizations. Following the verification and issuance of the materials, the corresponding invoice is cleared.

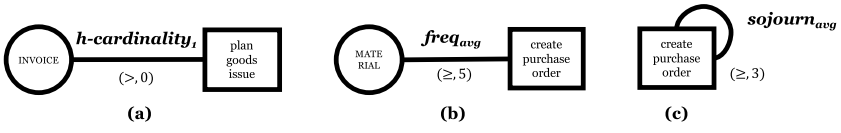


Fig. 12. OPGs representing the operational problems of the P2P process.

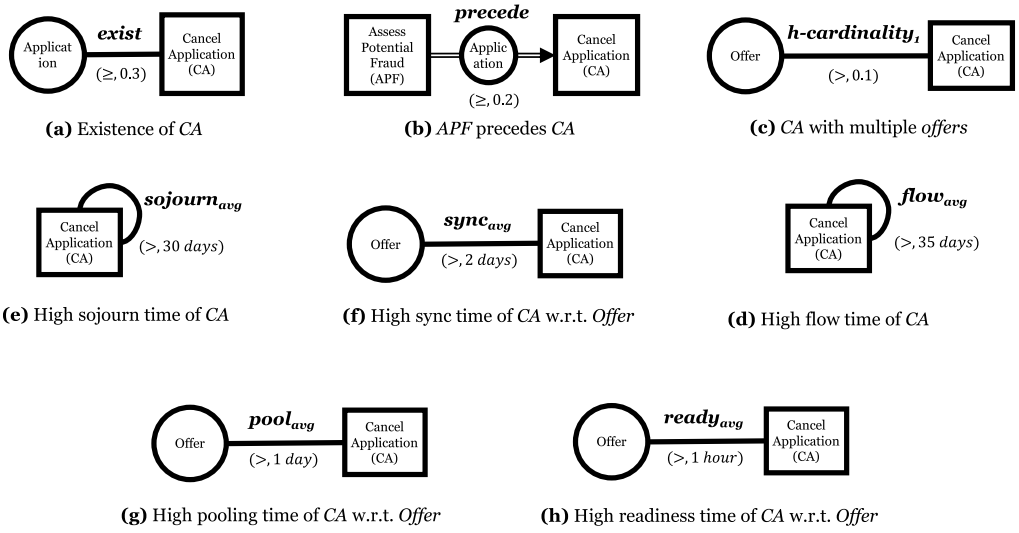
7.2. Use case using real-life data

We show a real-life use case to evaluate the effectiveness of the object-centric process monitoring and its sensitivity to the threshold values of problems and the sizes of event logs. To that end, we use event logs from a real-life loan application process of a Dutch financial institute[3] (van Dongen, 2017).

[3] The dataset in OCEL format (https://ocel-standard.org) is publicly available at https://github.com/gyunamister/proact/tree/master/samples/logs/bpi2017/

Table 3
Monitoring results of the P2P process. ✓ and ✗ denote the occurrence and non-occurrence of the problem, respectively.

Problems	Event Log					
	EL^1_{p2p}		EL^2_{p2p}		EL^3_{p2p}	
	Reality	*Monitoring*	*Reality*	*Monitoring*	*Reality*	*Monitoring*
Clearance of MIs	✓	✓	✓	✓	✗	✗
Excessive materials per PO	✗	✗	✗	✗	✓	✓
Delayed POC	✗	✗	✗	✗	✓	✓



(a) Existence of *CA* **(b)** *APF* precedes *CA* **(c)** *CA* with multiple *offers*

(e) High sojourn time of *CA* **(f)** High sync time of *CA* w.r.t. *Offer* **(d)** High flow time of *CA*

(g) High pooling time of *CA* w.r.t. *Offer* **(h)** High readiness time of *CA* w.r.t. *Offer*

Fig. 13. OPGs representing operational problems used in the experiments. Note that thresholds are set in the respective experiments.

Fig. 13 shows eight operational problems that were analyzed in this use case:

- *Existence of Cancel Application (CA)*: This problem identifies situations where cancellation actions within the application process occur with unusually high frequency, potentially indicating systemic issues or misuse.
- *Assess Potential Fraud (APF) Precedes CA*: This problem monitors scenarios where applications assessed for potential fraud are subsequently canceled. The goal is to prevent the wasteful use of resources spent on detailed assessments.
- *CA with Multiple Offers*: This metric flags a scenario where an application cancellation (CA) coincides with multiple offer considerations, suggesting indecision or conflicting process inputs, which could impact the efficiency of the loan processing.
- *High Flow Time of CA*: Monitors the duration from the initiation to the completion of a cancellation (CA), tracking any delays that could signify bottlenecks or inefficiencies within the cancellation process.
- *High Sojourn Time of CA*: Focuses on the total active processing time of cancellation (CA), exclusive of any wait or idle times, to identify and mitigate non-productive periods within the process.
- *High Sync Time of CA w.r.t. Offer*: Measures the synchronization time between cancellation (CA) and related offers, ensuring that all associated offers are handled in a synchronized manner such that the cancellation is processed in a timely manner.
- *High Pooling Time of CA w.r.t. Offer*: Examines the time that canceled applications require to collect all associated offers before final processing.
- *High Readiness Time of CA w.r.t. Offer*: Assesses the readiness time for processing cancellations once all offers have been finalized.

7.2.1. Effectiveness of object-centric process monitoring

To evaluate the effectiveness of object-centric process monitoring, we compare the results of our proposed approach with existing methods that are based on a single case notion. Many conventional approaches (cf. Section 2.1) can evaluate problems such as *Existence of CA*, *APF Precedes CA*, and *High Sojourn Time of CA*, with minor modifications.

In our analysis, we utilized a Declare-based approach to assess *Existence of CA* and *APF Precedes CA*, and applied commonly accepted metrics of sojourn time to evaluate *High Sojourn Time of CA* (van der Aalst, 2016).

Table 4 showcases the monitoring results for various problems from January 2016 to December 2017. Our proposed approach not only accurately detects all operational problems but also precisely calculates the associated metrics. While the existing methodologies correctly identify the *Existence of CA* and *High Sojourn Time of CA*, they reveal significant discrepancies in the measured sojourn times—25 days 13:58:03 compared to the actual 24 days 23:09:35. Furthermore, *APF Precedes CA* is not accurately detected by the existing approach, underscoring its limitations in recognizing and processing interactions between objects within events.

7.2.2. Sensitivity on threshold values

To evaluate the robustness of our proposed approach, we conducted a sensitivity analysis of threshold values across various operational problems. This analysis utilized 54 sub-event logs, each representing a week's worth of events, such as one log covering events from January 27, 2017, to February 3, 2017. By varying the threshold values within these logs, our objective is to examine how sensitive our monitoring method is to changes in this parameter. This helps in understanding the reliability of the approach under different threshold settings in a real-life scenario.

Fig. 14 shows the experimental results for the eight operational problems. For instance, the graph on the left-top corner describes the number of occurrences of *Existence of CA* by different threshold values ranging from 0.1 to 0.9. If the threshold is 0.1, the problem occurs in all 54 event logs, whereas the problem occurs in 50 event logs when the threshold is 0.2, etc.

Table 4
Monitoring results of the loan application process. ✓ and ✗ denote the occurrence and non-occurrence of the problem. The values in the parenthesis indicate the metrics computed for evaluating the problem occurrence.

Problems	Reality	Existing work	Our approach
Existence of CA	✓	✓ (0.3310)	✓ (0.3310)
APF precedes CA	✓	✗ (0.0001)	✓ (0.6690)
CA with multiple offers	✓	? (not available)	✓ (0.2450)
High flow time of CA	✗	? (not available)	✗ (27 days 14:21:49)
High sojourn time of CA	✗	✗ (25 days 13:58:03)	✗ (24 days 23:09:35)
High sync time of CA w.r.t. Offer	✓	? (not available)	✓ (2 days 00:23:45)
High pooling time of CA w.r.t. Offer	✓	? (not available)	✓ (1 day 17:10:17)
High readiness time of CA w.r.t. Offer	✗	? (not available)	✗ (00:25:44)

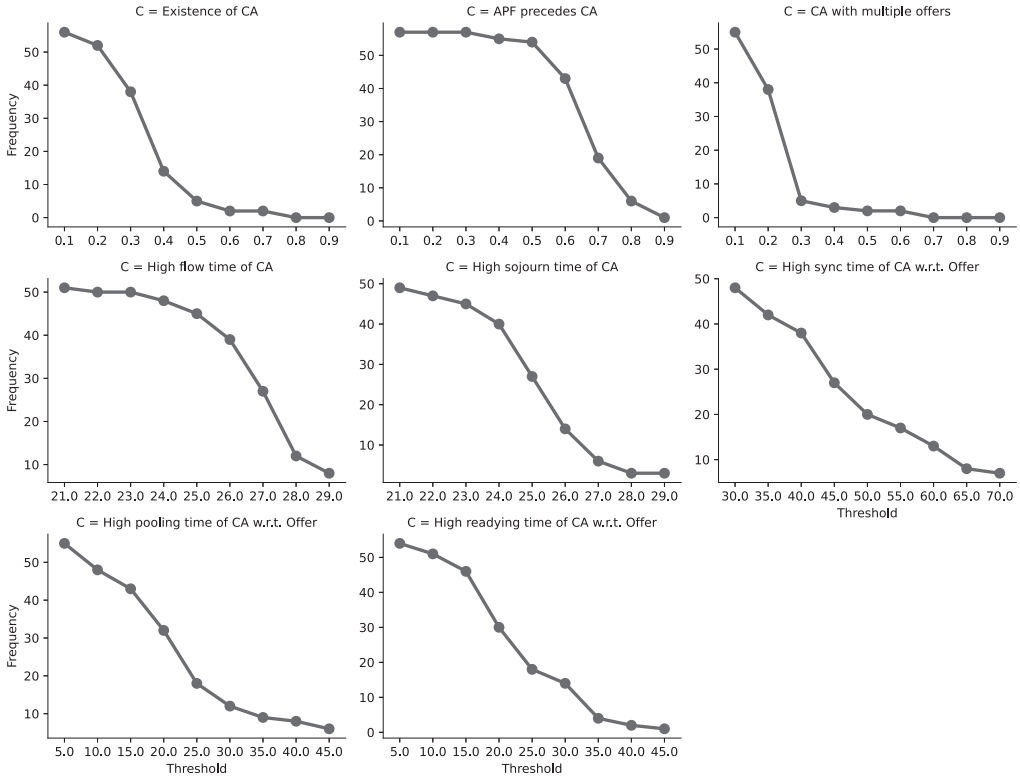


Fig. 14. Experimental results on the sensitivity of threshold values.

For all the problems, the number of occurrences significantly decreases in a certain range of threshold values. For instance, *Existence of CA* shows a significant decrease in the number of occurrences when the threshold value increases from 0.3 to 0.4. Moreover, the number of occurrences of *APF precedes CA* drops tremendously when the threshold value changes from 0.6 to 0.7 This implies that the number of occurrences is highly sensitive to the threshold value. Therefore, the threshold value should be considered carefully in the design of operational problems.

7.2.3. Sensitivity on monitoring time windows

Next, we investigate how the length of monitoring time windows affects the detection of operational problems. This analysis is essential for assessing the temporal sensitivity of our approach. To that end, we evaluate the occurrence of each problem depicted in Fig. 13 with a fixed threshold, adjusting the length of the monitoring time window where problems are evaluated. We set each problem's threshold value such that the problem occurs in 50 percent of the weekly event logs used in the previous experiment, i.e., 27 logs out of 54 sub-event logs. By extending the monitoring time window from 1 week to 5 weeks, we compute, for each problem, the ratio of event logs where the problem occurs (i.e., relative frequency).

Fig. 15 presents the experimental results pertaining to the sensitivity of the monitoring time windows. For example, the graph on the left-top corner indicates that the relative frequency of weekly event logs where *Existence of CA* occurs is 0.5, while the relative frequency of biweekly event logs where the problem occurs is around 0.45. The trend persists until the monitoring time window extends to 3 weeks, and it shifts when the monitoring time window reaches 4 weeks.

Although the relative frequency generally increases over the long term for most of the problems, this increase is not directly proportional to the length of the monitoring time window. For instance, the relative frequency of *High pooling time of CA w.r.t. offers* decreases when the monitoring time window extends from 3 weeks to 4 weeks. Moreover, *High readiness time of CA w.r.t. offers* decreases when the monitoring time window extends from 2 weeks to 3 weeks. Furthermore, some of the problems display more unstable trends. For example, the relative frequency of *CA with multiple offers* does not follow any specific trend, displaying irregular relative frequencies.

8. Discussion

Our use cases in both a manufacturing process and a Procure-To-Pay (P2P) process utilize simulated data to demonstrate the potential

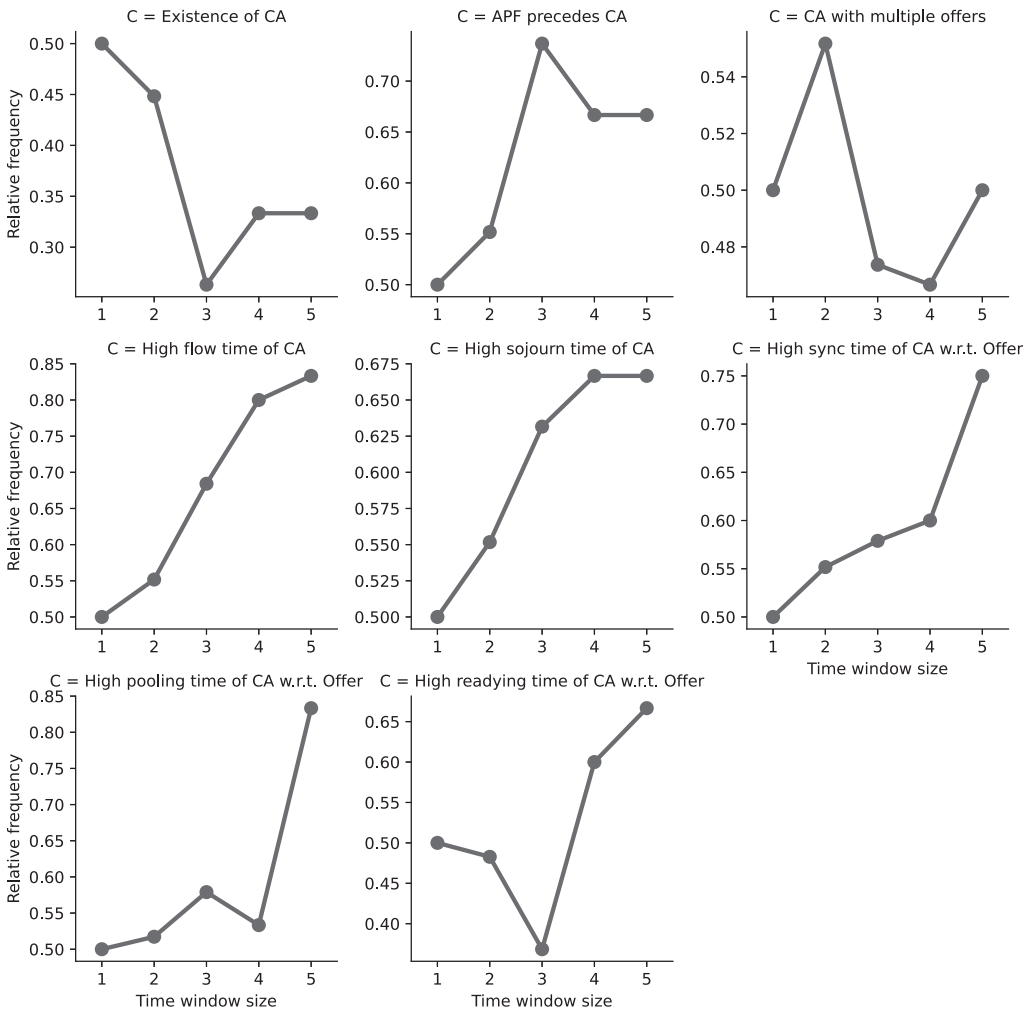


Fig. 15. Experimental results on the sensitivity of monitoring time windows.

applicability of the proposed approach in realistic scenarios. In both scenarios, we were able to successfully design operational problems corresponding to identified operational problems and assess their occurrence across different time windows. For instance, the ability to pinpoint issues such as bypassing purchase requisition approval and excessive reservations per production order within the manufacturing process demonstrates the strength of the proposed approach. Similarly, the P2P use case shows our method's capability to address diverse operational problems, such as the clearance of multiple invoices and excessive materials per Purchase Order (PO).

In our analysis of threshold value sensitivity, we found that the number of problem occurrences notably decreases within specific threshold ranges. This highlights that the detection of operational problems is highly sensitive to threshold values, emphasizing the importance of careful consideration in threshold setting. This is critical information for practitioners, as inappropriate thresholds could lead to the over- or under-detection of problems.

Our evaluation of the sensitivity of operational process monitoring over different monitoring time windows provides several intriguing insights. Firstly, the increase in the monitoring time window does not necessarily correlate with a uniform increase in the occurrence of problems. This is because of the relative nature of the metrics that define most of the problems. For instance, many problems use relative values such as the relative frequency to the whole events and the relative frequency to the whole objects. Therefore, expanding the monitoring time window does not result in a straightforward amplification of the occurrence of these problems.

Secondly, the increasing trend of problem occurrence with larger monitoring windows (e.g., for *High flow time of CA*) suggests periods of intensive problem occurrences. When certain problems occur intensively within a given period, they significantly influence the occurrence of problems within larger time windows. This may be due to specific operational activities or conditions during these periods, leading to the heightened incidence of problems. In contrast, the decreasing trend of problem occurrence with larger monitoring windows (e.g., for *Existence of CA*) is due to the dilution effect of the monitoring time window; as the window grows, the relative proportion of periods with lower problem occurrences increases, hence reducing the overall frequency of problem occurrence. This finding reinforces the importance of identifying and focusing on those periods with higher problem occurrences, even within larger monitoring time windows.

Our work can have important implications from both practical and academic viewpoints. First, the proposed approach in the study can significantly improve the effectiveness and accuracy of process mining and operational process monitoring in real-life business scenarios. By mitigating the issues associated with traditional event logs tied to a single case notion, practitioners can attain a more precise and comprehensive understanding of operational problems, leading to improved process efficiency, reduced operational risks, and better decision-making.

From an academic perspective, this study contributes a novel approach to process mining, addressing a significant gap in the literature related to event log issues. The development of the taxonomy of object-centric problems, OPGs, and object-centric process monitoring opens new avenues for further research. The object-centric perspective could

provide a more accurate model for understanding complex business processes, and future research could focus on refining these tools, exploring their applications in different contexts, or comparing their effectiveness with other methods in process mining.

However, this work has several limitations. First, the proposed OPGs can represent a comprehensive list of problems that cover diverse dimensions of business processes. However, the supported operational problems are by no means exhaustive. More advanced operational problems exist that are relevant to monitoring business processes. For instance, more advanced control-flow problems may include *multi choice*, *synchronization*, and *repetition* (Russell et al., 2016).

Next, some of the problems supported by the OPG are evaluated by focusing on a specific object type. For instance, a *co-existence* problem (e.g., activity *pick item* should not *co-exist* with activity *cancel item* w.r.t. an item) is evaluated by isolating an object type (e.g., item). However, more problems can be elicited when considering the interaction of different object types. For instance, activity *cancel item* w.r.t. an item should not *co-exist* with activity *send invoice* w.r.t. the corresponding order.

The recent development of a new standard for object-centric event logs (https://ocel-standard.org) introduces Object-To-Object (O2O) relations and qualifiers in both Event-To-Object (E2O) relations and O2O relations. These enhancements enable a deeper analysis of operational problems unaddressed by our current methods. Issues such as *supplier-product compatibility* and *supplier tier restrictions* in supply chain processes, which depend on the dynamics between multiple object types, suggest a need for approaches that can interpret and analyze these complex relationships.

Moreover, qualifiers in E2O and O2O relations can define advanced operational problems that our current approach does not address. In real-world scenarios, these relations are often characterized by qualifiers providing additional context, such as a *highly engaged* qualifier in an E2O relation indicating the importance of an object in relation to an event, or a *dependent* qualifier in an O2O relation signifying a hierarchical relationship between objects. Currently, the absence of mechanisms to incorporate such qualifiers in our object-centric problem graphs restricts our ability to analyze such complex operational problems.

While our sensitivity analysis provides valuable insights into how threshold values influence the detection of operational problems, our study does not offer explicit guidelines for setting these thresholds in practical scenarios, leaving it up to the judgement of domain experts. To address this gap, future research should focus on developing a framework that provides empirical guidelines for threshold determination based on a range of industry-specific case studies. Moreover, it is imperative to incorporate feedback mechanisms in the deployment of this approach, enabling continuous adjustment of thresholds based on real-world operational feedback and outcomes.

Finally, our design integrating rules and threshold logic within OPGs presents certain limitations in scenarios where thresholds may require frequent adjustments. A new threshold setting necessitates the creation of a new OPG, which can be cumbersome and inefficient. To mitigate this, an OPG management tool is required to enable users to easily adjust thresholds across multiple OPGs without altering the underlying rule logic. This tool is envisioned to support dynamic adjustments of thresholds based on seasonal variations, operational changes, or evolving business policies, thereby enhancing the practical applicability of our approach.

9. Conclusion

In this paper, we suggested a technique for object-centric operational process monitoring. We first introduced a list of object-centric operational problems. Next, we presented object-centric behavioral metrics, including control-flow metrics, object involvement metrics, and performance metrics. Afterward, we presented OPGs that represent the object-centric operational problems with formal semantics based on the object-centric behavioral metrics. Finally, we evaluated the approach by showing use cases in a manufacturing process and a P2P process supported by SAP ERP systems and by showing a real-life use case to evaluate the effectiveness of the proposed method and analyze the sensitivity of threshold values and the sensitivity of event log sizes using a real-life business process.

CRediT authorship contribution statement

Gyunam Park: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Wil M.P. van der Aalst:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have used a publicly available data, and the link to the data is provided in the manuscript.

Acknowledgments

We thank the Alexander von Humboldt (AvH) Stiftung, Germany for supporting our research.

References

Adams, J.N., Park, G., Levich, S., Schuster, D., van der Aalst, W.M.P., 2022. A framework for extracting and encoding features from object-centric event data. In: Troya, J., Medjahed, B., Piattini, M., Yao, L., Fernández, P., Ruiz-Cortés, A. (Eds.), Service-Oriented Computing - 20th International Conference, ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings. In: Lecture Notes in Computer Science, vol. 13740, Springer, pp. 36–53.

Adams, J.N., Park, G., van der Aalst, W.M.P., 2023. Preserving complex object-centric graph structures to improve machine learning tasks in process mining. Eng. Appl. Artif. Intell. 125, 106764.

Adams, J.N., van der Aalst, W.M.P., 2021. Precision and fitness in object-centric process mining. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (Eds.), ICPM 2021. pp. 128–135.

Awad, A., Barnawi, A., Elgammal, A., Shawi, R.E., Almalaise, A., Sakr, S., 2015. Runtime detection of business process compliance violations: an approach based on anti patterns. In: Wainwright, R.L., Corchado, J.M., Bechini, A., Hong, J. (Eds.), Proceedings of the 30th Annual ACM Symposium on Applied Computing. Salamanca, Spain, April 13-17, 2015, pp. 1203–1210.

Basin, D.A., Klaedtke, F., Müller, S., Zalinescu, E., 2015. Monitoring metric first-order temporal properties. J. ACM 62 (2), 15:1–15:45.

Berti, A., Park, G., Rafiei, M., van der Aalst, W.M.P., 2023. A generic approach to extract object-centric event data from databases supporting SAP ERP. J. Intell. Inf. Syst. 61, 835–857.

Berti, A., van der Aalst, W.M.P., 2019. Extracting multiple viewpoint models from relational databases. In: Ceravolo, P., van Keulen, M., López, M.T.G. (Eds.), SIMPDA 2018. In: Lecture Notes in Business Information Processing, vol. 379, pp. 24–51.

Berti, A., van der Aalst, W.M.P., 2023. OC-PM: analyzing object-centric event logs and process models. Int. J. Softw. Tools Technol. Transfer 25 (1), 1–17.

Bragaglia, S., Chesani, F., Mello, P., Montali, M., Sottara, D., 2011. Fuzzy conformance checking of observed behaviour with expectations. In: Pirrone, R., Sorbello, F. (Eds.), AI*IA 2011. In: Lecture Notes in Computer Science, vol. 6934, pp. 80–91.

Bruns, R., Dunkel, J., Offel, N., 2019. Learning of complex event processing rules with genetic programming. Expert Syst. Appl. 129, 186–199.

Burattin, A., Maggi, F.M., Sperduti, A., 2016. Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. 65, 194–211.

Burattin, A., van Zelst, S.J., Armas-Cervantes, A., van Dongen, B.F., Carmona, J., 2018. Online conformance checking using behavioural patterns. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (Eds.), BPM 2018. In: Lecture Notes in Computer Science, vol. 11080, pp. 250–267.

Chaudhry, N.A., Shaw, K., Abdelguerfi, M. (Eds.), 2005. Stream data management. In: Advances in Database Systems, vol. 30.
Cugola, G., Margara, A., 2012. Processing flows of information: From data stream to complex event processing. ACM Comput. Surv. 44 (3), 15:1–15:62.
Daum, M., Götz, M., Domaschka, J., 2012. Integrating CEP and BPM: how CEP realizes functional requirements of BPM applications (industry article). In: Bry, F., Paschke, A., Eugster, P.T., Fetzer, C., Behrend, A. (Eds.), DEBS 2012. pp. 157–166.
de Leoni, M., Marrella, A., 2017. Aligning real process executions and prescriptive process models through automated planning. Expert Syst. Appl. 82, 162–183.
Diba, K., Batoulis, K., Weidlich, M., Weske, M., 2020. Extraction, correlation, and abstraction of event data for process mining. WIREs Data Mining Knowl. Discov. 10 (3).
van Dongen, B., 2017. BPI challenge 2017.
Esser, S., Fahland, D., 2021. Multi-dimensional event data in graph databases. J. Data Semant. 10 (1–2), 109–141.
Galanti, R., Coma-Puig, B., de Leoni, M., Carmona, J., Navarin, N., 2020. Explainable predictive process monitoring. In: van Dongen, B.F., Montali, M., Wynn, M.T. (Eds.), 2nd International Conference on Process Mining. ICPM 2020, Padua, Italy, October 4-9, 2020, pp. 1–8.
Galanti, R., de Leoni, M., Navarin, N., Marazzi, A., 2023. Object-centric process predictive analytics. Expert Syst. Appl. 213 (Part), 119173.
Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P., 2021. OCEL: a standard for object-centric event logs. In: Bellatreche, L., Dumas, M., Karras, P., Matulevicius, R., Awad, A., Weidlich, M., Ivanovic, M., Hartig, O. (Eds.), New Trends in Database and Information Systems - ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Tartu, Estonia, August 24-26, 2021, Proceedings. In: Communications in Computer and Information Science, vol. 1450, Springer, pp. 169–175.
Hallé, S., Villemaire, R., 2008. Runtime monitoring of message-based workflows with data. In: IEEE ECOC 2008. pp. 63–72.
Hildebrandt, T.T., Mukkamala, R.R., Slaats, T., 2011. Nested dynamic condition response graphs. In: Arbab, F., Sirjani, M. (Eds.), FSEN 2011. In: Lecture Notes in Computer Science, vol. 7141, pp. 343–350.
Indiono, C., Mangler, J., Fdhila, W., Rinderle-Ma, S., 2016. Rule-based runtime monitoring of instance-spanning constraints in process-aware information systems. In: Debruyne, C., Panetto, H., Meersman, R., Dillon, T.S., Kühn, e., O'Sullivan, D., Ardagna, C.A. (Eds.), OTM 2016. In: Lecture Notes in Computer Science, vol. 10033, pp. 381–399.
Jensen, K., Kristensen, L.M., Wells, L., 2007. Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. Int. J. Softw. Tools Technol. Transfer 9 (3–4), 213–254.
Khayatbashi, S., Hartig, O., Jalali, A., 2023. Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In: ao Paulo A. Almeida, J., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (Eds.), Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, Portugal, November 6-9, 2023, Proceedings. In: Lecture Notes in Computer Science, vol. 14320, Springer, pp. 220–238.
Knuplesch, D., Reichert, M., Kumar, A., 2017. A framework for visually monitoring business process compliance. Inf. Syst. 64, 381–409.
Lee, W.L.J., Verbeek, H.M.W., Munoz-Gama, J., van der Aalst, W.M.P., Sepúlveda, M., 2018. Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. Inform. Sci. 466, 55–91.
Leitner, M., Mangler, J., Rinderle-Ma, S., 2012. Definition and enactment of instance-spanning process constraints. In: Wang, X.S., Cruz, I.F., Delis, A., Huang, G. (Eds.), WISE 2012. In: Lecture Notes in Computer Science, vol. 7651, pp. 652–658.
Li, G., de Carvalho, R.M., van der Aalst, W.M.P., 2017. Automatic discovery of object-centric behavioral constraint models. In: Abramowicz, W. (Ed.), Business Information Systems - 20th International Conference, BIS 2017, Poznan, Poland, June 28-30, 2017, Proceedings. In: Lecture Notes in Business Information Processing, vol. 288, pp. 43–58.
Liss, L., Adams, J.N., van der Aalst, W.M.P., 2023. Object-centric alignments. In: ao Paulo A. Almeida, J., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (Eds.), Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, Portugal, November 6-9, 2023, Proceedings. In: Lecture Notes in Computer Science, vol. 14320, Springer, pp. 201–219.
Ly, L.T., Rinderle-Ma, S., Dadam, P., 2010. Design and verification of instantiable compliance rule graphs in process-aware information systems. In: Pernici, B. (Ed.), CAiSE 2010. In: Lecture Notes in Computer Science, vol. 6051, pp. 9–23.
Maggi, F.M., Westergaard, M., Montali, M., van der Aalst, W.M.P., 2011. Runtime verification of LTL-based declarative process models. In: Khurshid, S., Sen, K. (Eds.), RV 2011. pp. 131–146.
Montali, M., Maggi, F.M., Chesani, F., Mello, P., van der Aalst, W.M.P., 2013. Monitoring business constraints with the event calculus. ACM Trans. Intell. Syst. Technol. 5 (1), 17:1–17:30.
Park, G., Adams, J.N., van der Aalst, W.M.P., 2022. OPerA: Object-centric performance analysis. In: Ralyté, J., Chakravarthy, S., Mohania, M., Jeusfeld, M.A., Karlapalem, K. (Eds.), ER 2022. In: Lecture Notes in Computer Science, vol. 13607, pp. 281–292.
Park, G., van der Aalst, W.M.P., 2022. Monitoring constraints in business processes using object-centric constraint graphs. In: Montali, M., Senderovich, A., Weidlich, M. (Eds.), Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers. In: Lecture Notes in Business Information Processing, vol. 468, Springer, pp. 479–492.
Pesic, M., Schonenberg, H., van der Aalst, W.M.P., 2007. DECLARE: full support for loosely-structured processes. In: EDOC 2007. pp. 287–300.
Ramezani, E., Fahland, D., van der Aalst, W.M.P., 2012. Where did I misbehave? Diagnostic information in compliance checking. In: Barros, A., Gal, A., Kindler, E. (Eds.), BPM 2012. In: Lecture Notes in Computer Science, vol. 7481, pp. 262–278.
Rebmann, A., Rehse, J., van der Aa, H., 2022. Uncovering object-centric data in classical event logs for the automated transformation from XES to OCEL. In: Ciccio, C.D., Dijkman, R.M., del-Río-Ortega, A., Rinderle-Ma, S. (Eds.), Business Process Management - 20th International Conference, BPM 2022, MÜNster, Germany, September 11-16, 2022, Proceedings. In: Lecture Notes in Computer Science, vol. 13420, Springer, pp. 379–396.
Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., 2016. Workflow Patterns: The Definitive Guide.
Santos, E.A.P., Francisco, R., Vieira, A.D., Loures, E.D.F.R., de Paula, M.A.B., 2011. Modeling business rules for supervisory control of process-aware information systems. In: Daniel, F., Barkaoui, K., Dustdar, S. (Eds.), BPM 2011 International Workshops. In: Lecture Notes in Business Information Processing, vol. 100, pp. 447–458.
dos Santos Garcia, C., Meincheim, A., Junior, E.R.F., Dallagassa, M.R., Sato, D.M.V., Carvalho, D.R., Santos, E.A.P., Scalabrin, E.E., 2019. Process mining techniques and applications - A systematic mapping study. Expert Syst. Appl. 133, 260–295.
Stefanowski, J., Cuzzocrea, A., Slezak, D., 2014. Processing and mining complex data streams. Inform. Sci. 285, 63–65.
Türetken, O., Elgammal, A., van den Heuvel, W., Papazoglou, M.P., 2012. Capturing compliance requirements: A pattern-based approach. IEEE Softw. 29 (3), 28–36.
van der Aalst, W.M.P., 2015. Business process simulation survival guide. In: vom Brocke, J., Rosemann, M. (Eds.), Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, second ed. In: International Handbooks on Information Systems, Springer, pp. 337–370.
van der Aalst, W.M.P., 2016. Process Mining - Data Science in Action, second ed.
van der Aalst, W.M.P., 2019. Object-centric process mining: Dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (Eds.), SEFM 2019. pp. 3–25.
van der Aalst, W.M.P., 2020. Academic view: Development of the process mining discipline. In: Reinkemeyer, L. (Ed.), Process Mining in Action: Principles, Use Cases and Outlook. Cham, pp. 181–196.
van der Aalst, W.M.P., 2023. Object-Centric Process Mining: The next frontier in business performance. White Paper, Celonis, pp. 1–24.
van der Aalst, W.M.P., Berti, A., 2020. Discovering object-centric Petri nets. Fund. Inform. 175 (1–4), 1–40.
Weidlich, M., Ziekow, H., Gal, A., Mendling, J., Weske, M., 2014. Optimizing event pattern matching using business process models. IEEE Trans. Knowl. Data Eng. 26 (11), 2759–2773.
Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N., 2011. Event-based monitoring of process execution violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (Eds.), BPM 2011. In: Lecture Notes in Computer Science, vol. 6896, pp. 182–198.
Winter, K., Rinderle-Ma, S., 2017. Discovering instance-spanning constraints from process execution logs based on classification techniques. In: Hallé, S., Villemaire, R., Lagerström, R. (Eds.), EDOC 2017. pp. 79–88.
Xiong, J., Xiao, G., Kalayci, T.E., Montali, M., Gu, Z., Calvanese, D., 2022. A virtual knowledge graph based approach for object-centric event logs extraction. In: Montali, M., Senderovich, A., Weidlich, M. (Eds.), Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers. In: Lecture Notes in Business Information Processing, vol. 468, Springer, pp. 466–478.
van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., van der Aalst, W.M.P., 2019. Online conformance checking: relating event streams to process models using prefix-alignments. Int. J. Data Sci. Anal. 8 (3), 269–284.