Precision and Fitness in Object-Centric Process Mining

Jan Niklas Adams
Chair of Process and Data Science
RWTH Aachen University
Aachen, Germany
niklas.adams@pads.rwth-aachen.de

Wil M.P. van der Aalst
Chair of Process and Data Science
RWTH Aachen University
Aachen, Germany
wvdaalst@pads.rwth-aachen.de

***Abstract*—Traditional process mining considers only one single case notion and discovers and analyzes models based on this. However, a single case notion is often not a realistic assumption in practice. Multiple case notions might interact and influence each other in a process. Object-centric process mining introduces the techniques and concepts to handle multiple case notions. So far, such event logs have been standardized and novel process model discovery techniques were proposed. However, notions for evaluating the quality of a model are missing. These are necessary to enable future research on improving object-centric discovery and providing an objective evaluation of model quality. In this paper, we introduce a notion for the precision and fitness of an object-centric Petri net with respect to an object-centric event log. We give a formal definition and accompany this with an example. Furthermore, we provide an algorithm to calculate these quality measures. We discuss our precision and fitness notion based on an event log with different models. Our precision and fitness notions are an appropriate way to generalize quality measures to the object-centric setting since we are able to consider multiple case notions, their dependencies and their interactions.**

***Keywords*—Object-Centric Process Mining, Precision, Fitness**

I. Introduction

In the past years, process mining [1] has introduced a data-driven way to discover and analyze processes. The introduced techniques and algorithms enable organizations to discover the true underlying process from the data of its execution rather than manually designing what is assumed to be the process.

These techniques work on an event log which records the executed activities for different cases. Each event has an activity and is attributable to exactly one case, e.g., the customer, order, patient, etc.. The attribute used to assign an event to a case is called the case notion. The event log, therefore, describes in which order the activities can be executed for such cases.

However, this already shows the central limitation of traditional process mining: the single case notion. In practice, the assumption that every event is only related to exactly one case is unrealistic. One such example could be an event describing activity *Load cargo* for objects *plane1* and *bag1* at *2021-10-02 17:23*. If one considers this event and the corresponding event log, what would be the case notion? The plane to be loaded with cargo or the bag to be loaded into the plane? By focusing on only one case notion one would omit the behaviour of the other case notions, e.g., focusing on only the plane would omit the process of checking in and transporting baggage. Uniting the different case notions into one single, comprehensive process model allows us to consider dependencies between the executions that are not available otherwise.

Object-centric process mining [2] addresses these problems and aims to define techniques and standards for retrieving and analyzing comprehensive, easy-to-understand models of processes with multiple case notions. An event log format [3] and a first discovery technique [4] have already been introduced. This discovery technique yields an object-centric Petri net. However, the missing key to enable further research on object-centric process discovery on the one hand and objective model evaluation on the other hand are quality criteria that link object-centric model and log and specify the conformance of the model. In traditional process mining quality criteria like precision, fitness, simplicity and generality are used to express the quality of a model with respect to a log [1]. These can be used to compare different models of the same event log, evaluate the results of different discovery algorithms, specify the conformance of a handmade process model and more. So far, there are no quality measures to evaluate an object-centric Petri net. To facilitate further research we need a way to describe the quality of an object-centric Petri net, e.g., how well it fits the data or how precisely it describes the data. Due to multiple case notions and many-to-many relationships fitness and precision notions from traditional process mining can not trivially be adapted to the object-centric setting.

In this paper, we introduce a fitness and precision notion for object-centric Petri nets with respect to object-centric event logs. Our fitness notion can be seen as an object-centric adaption to replaying traces [5], the precision notion as a generalization of the escaping edges precision [6].

The paper is structured as follows. In Section II, we discuss other approaches of handling multiple case notions. In Section III, we introduce object-centric event logs and Petri nets and illustrate them on the basis of a running example. In Section IV, we formally introduce our fitness and precision notion. This is further accompanied by our running example. In Section V, we present an algorithm to calculate both

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

precision and fitness. We use three different models with respect to an event log in Section VI to evaluate our introduced notions. We conclude this paper in Section VII.

II. Related Work

In this section, we introduce the related work on handling multiple case notions and the techniques to determine fitness and precision in traditional process mining.

The different approaches to handle multiple case notions can be grouped into three categories: interacting processes, colored Petri net approaches, and novel modeling techniques. We discuss the corresponding approaches, their disadvantages and how object-centric Petri nets relate to these.

Several approaches to handle multiple case notions use individual processes with some notion for interaction between them. The first method that was introduced is proclets [7], describing interacting workflow processes. Over time, this concept was extended to cover many-to-many relationships [8] and different analysis techniques [9]. Another modeling technique is artifact-centric modeling [10], [11]. Process discovery techniques for artifact-centric modeling have been developed [12], as well as conformance checking methods [13], [14]. To deal with the complexity of artifact-centric modeling, [15] introduced a restricted artifact-centric process model definition where no concurrency is allowed within one artifact. The main disadvantage of these approaches is the absence of a single comprehensive model of the process as they show interacting individual processes. Furthermore, models tend to get too complex and hard to understand.

The second approach to handle multiple case notions are colored Petri nets. DB-nets [16] introduce a modeling technique to include a data-perspective into a process model using a colored Petri net. However, the modeling is hard to understand for a user and it targeted to a modeling, not a mining setting. Furthermore, there is a plethora of approaches that uses colored Petri nets with some restrictions, e.g., no concurrency within one case notion and discarding variable one-to-many relationships, which restrictions render them infeasible in many settings. The discussion of these approaches is outside the scope of this paper.

The third group includes newly proposed modeling techniques to condense a process with multiple case notions to one model. Object-Centric Behavioral Constraints (OCBC) [17] are a recently introduced approach to show behavior and relationships of different objects in one model. Discovery algorithms as well as quality measures have been proposed for this discovery technique [18]. However, since OCBC builds on top of a data format that records the whole evolution of a database the models tend to get complex and the proposed techniques are not scalable. Multiple View Point models [19] introduce less complex models where the discovery is more scalable. A process model can be constructed by correlating events via objects. [4] extends this approach to discover object-centric Petri nets, Petri nets with places of different colors and arcs that can consume a variable amount of tokens. Object-centric Petri nets can model a process with multiple case notions in one single model, consider one-to-many and many-to-many relations, concurrency within case notions and a scalable discovery algorithm is available. Therefore, object-centric Petri nets alleviate most of the drawbacks from other approaches in the related work with respect to multiple case notions.

Since quality metrics play an important role in traditional process mining several techniques to determine fitness and precision have been proposed. Techniques for determining fitness include causal footprints, token-based replay, alignments, behavioral recall and eigenvalue recall [20]. In this paper, we use an adaptation of a replaying fitness, i.e., whether the preoccurring activities of an event can be replayed on the object-centric Petri net. Techniques for determining precision include escaping edges precision, negative event precision and projected conformance checking [21]. The adaptation of these techniques to object-centric event logs and Petri nets each pose their own challenges. i.e., dealing with multiple case notions and variable arcs in the Petri net. In this paper, we propose a generalization of the escaping edges precision [6] which links an event to a state in the process model and determines the behavior allowed by model and log and, subsequently, the precision. This can also be seen as an object-centric adaptation of replaying history on the process model to determine fitness and precision [22].

III. Object-Centric Process Mining

Object-centric process mining moves away from the single case notion of traditional process mining, i.e., every event is related to exactly one case, and opens up the possibility for one event being related to multiple, different case notions. The foundations were defined in [4]. We introduce these key concepts of object-centric process mining in this section. These are accompanied by a running example of a flight process that considers planes and how the associated baggage is handled. It describes the operations of the plane, i.e., fueling, cleaning and lift off, and the handling of baggage, i.e., check-in, loading and pick up. We first introduce some basic definitions and notations.

Definition 1 (Power Set, Multiset, Sequence)**.** *Let X denote a set. $\mathcal{P}(X)$ denotes the power set of these elements, the set of all possible subsets. A multiset $B: X \to \mathbb{N}$ assigns a frequency to each element in the set and is denoted by $[x^m]$ for $x \in X$ and frequency $m \in \mathbb{N}$. A sequence of length n assign positions to elements $x \in X$ of a set $\sigma: \{1, \ldots, n\} \to X$. It is denoted by $\sigma = \langle x_1, \ldots, x_n \rangle$. The concatenation of two sequences is denoted with $\sigma_1 \cdot \sigma_2$. Concatenation with an empty sequence does not alter a sequence $\sigma \cdot \langle\rangle = \sigma$. The number of elements in a sequence σ is given by $len(\sigma)$.*

The central part of object-centric process mining are objects. Each object is of exactly one object type.

Definition 2 (Object and Object Types)**.** *Let A be the universe of activities. Let U_o be the universe of objects and U_{ot} be the universe of object types. The function $otyp: U_o \to U_{ot}$ maps*

TABLE I
EXAMPLE OF AN OBJECT-CENTRIC EVENT LOG L_1

Event ID	Activity	*plane*	*baggage*
e_1	*Fuel plane*	*p1*	
e_2	*Check-in*		*b1*
e_3	*Check-in*		*b2*
e_4	*Load cargo*	*p1*	*b1,b2*
e_5	*Lift off*	*p1*	
e_6	*Unload*	*p1*	*b1,b2*
e_7	*Pick up @ dest*		*b1*
e_8	*Pick up @ dest*		*b2*
e_9	*Clean*	*p1*	
e_{10}	*Fuel plane*	*p2*	
e_{11}	*Check-in*		*b3*
e_{12}	*Check-in*		*b4*
e_{13}	*Load cargo*	*p2*	*b3,b4*
e_{14}	*Lift off*	*p2*	
e_{15}	*Unload*	*p2*	*b3,b4*
e_{16}	*Clean*	*p2*	
e_{17}	*Pick up @ dest*		*b3*
e_{18}	*Pick up @ dest*		*b4*

each object to its type.

The two universes, of objects and of object types, contain all possible objects and all possible object types. Every object $o \in U_o$ has a type $_{otyp}(o)$. In our example we are considering object types $OT = \{planes; baggage\}$. We have, in total, six objects: $O = \{p1; p2; b1; b2; b3; b4\}$ of which two are planes and four are baggage, indicated by the first letter of the object, i.e., $_{otyp}(p1) =$ $_{otyp}(p2) = plane$; $_{otyp}(b1) =$ $= _{otyp}(b4) = baggage$. In general, we are interested in recording the behaviour of objects over time, i.e., at which point in time activities were executed that concern objects. This is done using an object-centric event log. The OCEL format [3] records the corresponding activity, timestamp and objects of each object type for an event. Additional attributes are stored for each object. We use a reduced definition of the object-centric event log to focus on the aspects relevant for this paper.

Definition 3 (Object-Centric Event Log)**.** *Let U_E be the universe of event identifiers. An object-centric event log is a tuple $L = (E; OT; O;$ $_{act};$ $_{omap};$ $)$ of event identifiers E U_E, object types OT U_{ot}, objects O U_o, a mapping function from event to activity $_{act}: E \not\to A$, a mapping function from event to related objects $_{omap}: E \not\to P(O)$ and a total order .*

This definition gives us event identifiers of which each is related to an activity and to some objects. These event identifiers are subject to a total order, i.e., by the time of their occurrence. An example of an object-centric event log is depicted in Table I. It consists of the events with the identifiers $E = \{e_1; : : : ; e_{18}\}$. The event identifier is unique and provides the order of events. The activity of an event, e.g., $_{act}(e_4) =$ *Load cargo* is given as well as the related objects, e.g., $_{omap}(e_4) = \{p1,b1,b2\}$. Discovering process models from event data plays a central role in process mining as it is a data-driven way to uncover the true nature of a process. One way to represent process models are Petri nets [1]. Van der Aalst and Berti [4] introduce the concept of object-centric Petri nets and provide an algorithm to discover an object-centric Petri net from an object-centric event log. An object-centric Petri net is a colored Petri net where the places are colored as one of the object types and some arcs can consume a variable amount of tokens.

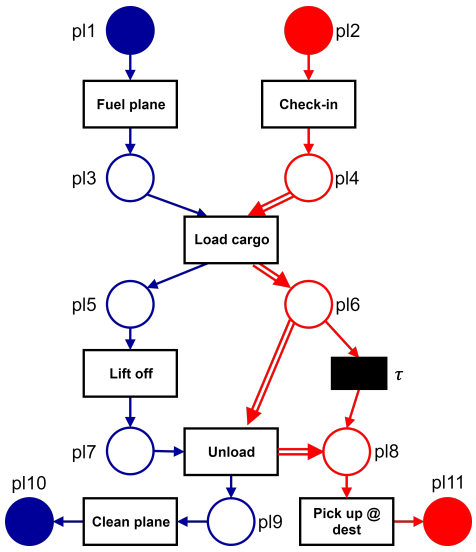

Fig. 1. Example of an object-centric Petri net $OCPN_1$.

Definition 4 (Object-Centric Petri Net)**.** *Let $N = (P; T; F; l)$ be a Petri net consisting of places P, transitions T, flow relations F $(T$ $P) [(P$ $T)$ and a labeling function $l: T \not\to A$. An object-centric Petri net is a tuple $OCPN = (N; pt; F_{var})$ of Petri net N, a mapping of places to object types $pt: P \not\to OT$ and a set of variable arcs F_{var} F. We introduce the notations:*

- *$t = \{p \in P \mid (p; t) \in F\}$ is the preset of a transition t.*
- *$t^\bullet = \{p \in P \mid (t; p) \in F\}$ is the postset of a transition t.*
- *$tpl(t) = \{pt(p) \mid p \in t [t^\bullet\}$ are the object types associated to a transition t.*
- *$tpl_{nv}(t) = \{pt(p) \mid p \in P \wedge \{(p; t); (t; p)\} \setminus (F \cap F_{var}) \neq \emptyset\}$ are the non-variable object types associated to transition t.*

An example of an object-centric Petri net for the flight process is depicted in Figure 1. Places are indicated by circles and transitions are indicated by rectangles. Arcs are indicated by arrows while variable arcs are indicated by double arrows. The object type of a place is shown by its color, blue being type *plane* and red being type *baggage*. The label of each transition is depicted inside the transition itself. The transition with no label is a so called silent transition. We refer to this transition as . This definition of the Petri net itself is not sufficient to describe a process and the behavior it allows. We need to introduce further concepts. We, first, introduce the concept of a marking. Places of object-centric Petri nets can hold tokens of the same object type. The marking of an object-centric Petri net can be expressed as objects being associated to places of the corresponding object type.

Definition 5 (Marking of an Object-Centric Petri Net)**.** *Let $OCPN = (N; pt; F_{var})$ be an object-centric Petri net with*

$N = (P, T, F, l)$. *The set of possible tokens is described by* $Q_{OCPN} = \{(p, o) \in P \times U_o \mid pt(p) = otyp(o)\}$. *A marking is a multiset of tokens* $M \in B(Q_{OCPN})$.

A marking describes the current state of the Petri net. It states which objects are contained in which places. We can move between markings by firing transitions. The concept of a binding is used to describe this. A binding is a tuple of a transition and the involved objects of every object type. The objects of the corresponding color in the binding are consumed in the input places and produced in the output places. A binding is only enabled at a given marking if each input place of the corresponding transition holds at least the tokens of this object type contained in the binding.

Definition 6 (Binding Execution)**.** *Let* $OCPN = (N, pt, F_{var})$ *be an object-centric Petri net with* $N = (P, T, F, l)$. $B = \{(t, b) \in T \times (U_{ot} \not\to P(U_o)) \mid dom(b) = tpl(t) \wedge \forall_{ot \in tpl_{nv}(t)} |b(ot)| = 1\}$ *defines the set of all possible bindings. The consumed tokens of a binding* $(t, b) \in B$ *are defined by* $cons(t, b) = [(p, o) \in Q_{OCPN} \mid p \in t \wedge o \in b(pt(p))]$ *and the produced tokens are defined by* $prod(t, b) = [(p, o) \in Q_{OCPN} \mid p \in t \wedge o \in b(pt(p))]$. *A binding* $(t, b) \in B$ *in marking* M *is enabled if* $cons(t, b) \subseteq M$. *The execution of an enabled binding in marking* M *leads to marking* $M' = M - cons(t, b) + prod(t, b)$. *This is denoted by* $M \xrightarrow{(t,b)} M'$

$M \xrightarrow{(t,b)} M'$ indicates that the occurrence of binding (t, b) in marking M leads to marking M'. Multiple subsequent enabled bindings can be encoded as sequence $\sigma = \langle (t_1, b_1), \ldots, (t_n, b_n) \rangle \in B^*$. A sequence that starts in marking M_0 and results in marking M_n is encoded as $M_0 \xrightarrow{\sigma} M_n$. We can take a binding at the *Load cargo* transition of the Petri net in Figure 1 as an example. We assume a marking where the input places of *Load cargo* contain plane $p1$ and baggages $b1, b2$ according to their color $[(pl3, p1), (pl4, b1), (pl4, b2)]$. In this state of the Petri net the binding of *Load cargo* and $p1, b1, b2$ is enabled since the input places of the transition contain enough tokens of objects contained in the binding. The binding can be executed and leads to a new marking where the tokens of $p1, b1, b2$ moved from the input to the output places of *Load cargo*. We can construct a sequence of such enabled bindings that describes the execution of several bindings after each other. In this way, we can express that one marking is reachable from another marking in the Petri net. We define initial and final markings. Together with the object-centric Petri net these form an accepting object centric Petri net. All binding sequences starting from an initial marking and ending in a final marking form the accepted behavior of that accepting object-centric Petri net.

Definition 7 (Accepting Object-Centric Petri Net)**.** *Let* $OCPN = (N, pt, F_{var})$ *be an object-centric Petri net with* $N = (P, T, F, l)$ *and let* $M_{init}, M_{final} \in B(Q_{OCPN})$ *be initial and final markings of that Petri net.* $OCPN_A = (OCPN, M_{init}, M_{final})$ *is a tuple that describes an accepting object-centric Petri net.* $= \{\sigma \in B^* \mid M \in M_{init} \wedge M' \in M_{final} \wedge M_{init} \xrightarrow{\sigma} M_{final}\}$ *describes all the binding sequences, i.e., the accepted behavior of the Petri net.*

In our example, an accepted behavior is a binding sequence that starts in a marking with only tokens in the places $pl1$ and $pl2$ and ends in a marking with only tokens in the places $pl10$ and $pl11$. The algorithm of [4] can be used to discover such accepting object-centric Petri nets.

These concepts are sufficient to describe the behavior allowed by an object-centric Petri net. Starting in an initial marking all binding sequences are allowed that lead to a final marking. However, if we consider our event log in Table I and our model in Figure 1 the question arises of how well the process model describes the object-centric event log. So far research on object-centric process mining has focused on specifying standardized event log formats and discovering models. The missing key to enable further research on improving object-centric process discovery are quality criteria for a discovered object-centric model.

IV. Fitness and Precision for Object-Centric Petri Nets

Generally speaking, fitness and precision compare the behaviour seen in a data set to the behaviour possible in a model. The more behavior of the data set the model allows the fitter it is. The more behaviour the model allows that is not covered by the data set the more imprecise it is. This general notion of fitness and precision can be adapted to event logs and process models by comparing the recorded activity sequences in the event log to the possible activity sequences in the process model [22] in traditional process mining. However, this can not easily be adapted to object-centric event logs and Petri nets. The main problem is multiple case notions of an event. There is no single sequence of previously occurring activities for an event anymore. Multiple involved objects are associated to different activity sequences that were previously executed for them. Furthermore, even if there is only one object involved in an event, an activity sequence of one object might be dependent on the occurrence of an activity sequence of another object. Compare the exemplary event log in Table I. Before event e_5, *Lift off* of plane $p1$, happens, the baggages $b1$ and $b2$ have to be loaded into the plane, making this event also dependent on these two objects even though they are not included in the event itself. We define a context notion that describes the previously executed activity sequences of the objects on which an event depends. The dependent objects and their relevant events are extracted by constructing the event-object graph.

Definition 8 (Event-Object Graph and Context of an Event)**.** *Let* $L = (E, OT, O, \pi_{act}, \pi_{omap}, \ldots)$ *be an object-centric event log, let* $ot \in OT$ *be an object type and let* $o \in O$ *be an object. We introduce the following notations:*

$G_L = (V, K)$ *is the event-object graph of the event log* L *with* $V = E$ *and* $K = \{(e', e) \in E \times E \mid e' <_e \wedge \pi_{omap}(e) \cap \pi_{omap}(e') \neq \emptyset\}$.

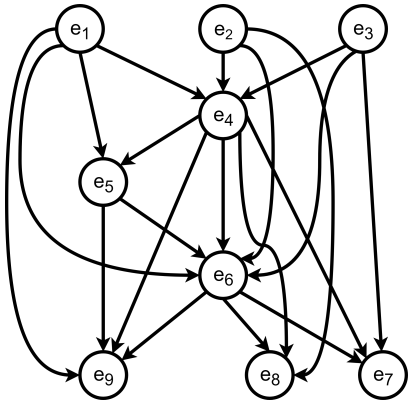

Fig. 2. Event-object graph of events $\{e_1, \ldots, e_9\}$ of log L_1

$e = \{e' \in V \mid \exists\, \sigma = \langle e^0, \ldots, e \rangle \in E^* \; \forall i \in \{1, \ldots, |\sigma|\}\ (\sigma_i, \sigma_{i+1}) \in K\}$ *is the event preset of an event* $e \in E$.

$e\#_o = \langle \pi_{act}(e_1), \ldots, \pi_{act}(e_n) \rangle$ *such that*

$\{e' \in \sigma \mid e' \in \pi_{omap}(e^0)\} = \{e_1, \ldots, e_n\}$

$e_1 < \ldots < e_n$

The context of an event is defined as $context_e(ot) = [\, e\#_o \mid o \in \bigcup_{e' \in \sigma \wedge e \in \pi_{omap}(e')} \pi_{otyp}(o) = ot]$ *for any* $ot \in OT$.

Analogously to the single case notion, each single object has a simple sequence of its previously occurring activities. We use a multiset of sequences to cover the possibility of an event being related to multiple objects of the same object type with the same sequence. We use the event-object graph to grasp every object on which the execution of an event depends and up to which event it depends on this object. The event-object graph introduces a directed edge between events if they share an object. For an event e, all other events that have a directed path to event e form the event preset of e, the events on which the execution of e depends. The full event-object graph of events e_1 to e_9 of log L_1 is depicted in Figure 2. Take event e_5 as an example. The event preset of e_5 is formed by all events that are directly or transitively connected to e_5, i.e., that have a directed path the e_5 in the event-object graph. This includes e_1 and e_4 but also e_2 and e_3 even though they do not share any objects with e_5. Therefore, $e_5 = \{e_1, e_2, e_3, e_4\}$. This shows that the event preset includes all events on which the execution of an event depends, also the transitive dependencies. To construct the context we take all objects that appear in the event preset and the event itself, map them onto their sequence of activities in the event preset and construct a multiset of the objects' sequences for each object type. For e_5 this includes the objects $p1, b1, b2$. We construct the sequences of these and combine them to the context $context_{e_5}(plane) = [\langle$*Fuel plane, Load cargo*$\rangle]$; $context_{e_5}(baggage) = [\langle$*Check-in, Load cargo*$\rangle^2]$

The context can colloquially be described as everything that had to happen for this event to occur. We use the context to link log and model behaviour. The context specifies several sequences of activities executed for different object types. A binding sequence of an object-centric Petri net, thus, also has a context if we consider the executed activity sequences of all the objects in this binding sequence.

Definition 9 (Context of a Binding Sequence)**.** *Let* $OCPN_A = (OCPN, M_{init}, M_{final})$ *be an accepting object-centric Petri net with* $OCPN = (N, pt, F_{var})$ *and* $N = (P, T, F, l)$. *Let* $\sigma = \langle (t_1, b_1), \ldots, (t_n, b_n) \rangle$ *be a sequence of enabled bindings of that Petri net. We introduce the following notations:*

$(t, b)\#_o = \langle l(t) \rangle$ *if* $o \in b(\pi_{typ}(o)) \wedge t \in dom(l)$ *otherwise* $(t, b)\#_o = \langle\rangle$.

$\sigma\#_o = (t_1, b_1)\#_o \cdot \ldots \cdot (t_n, b_n)\#_o$ *is the prefix of an object for a binding sequence.*

context $(ot) = [\, \sigma\#_o \mid o \in \bigcup_{i \in \{1, \ldots, n\}} b_i(ot)]$ *for any* $ot \in U_{ot}$ *is the multiset of prefixes in a binding sequence, also called context of the binding sequence.*

Given a binding sequence, we can project it onto the transition labels for each object that is included in at least one of the bindings. We do this by projecting each binding of the sequence onto its transition label if the object is contained in the binding. If the transition is a silent transition, i.e., it has no label, this projection yields the empty sequence. The labels are concatenated into a sequence of labels. This is the prefix for an object. The prefixes are united into a multiset for each object type. To illustrate this we use the Petri net from Figure 1 and a binding sequence. In our example, a binding consists of a transition name and bounded objects b of each object type: $\sigma_1 = \langle$(*Fuel Plane*; b(*plane*) = $\{p1\}$; b(*baggage*) = $\{\}$); (*Check-in*; b(*plane*) = $\{\}$; b(*baggage*) = $\{b1\}$); (*Check-in*; b(*plane*) = $\{\}$; b(*baggage*) = $\{b2\}$); (*Load cargo*; b(*plane*) = $\{p1\}$; b(*baggage*) = $\{b1, b2\}$)$\rangle$. All objects that appear in this sequence are $\{p1, b1, b2\}$. The projected prefix for *p1* is $\langle$*Fuel plane, Load cargo*$\rangle$. For each *b1* and *b2* this sequence is $\langle$*Check-in, Load cargo*$\rangle$. The context for this binding sequence is, therefore, $context_{\sigma_1} = context_{e_5}$.

Given a certain context one can look at all the possible binding sequences that have this context. All of the states that are reached after executing any of these binding sequences are the states that are reachable given this context.

Definition 10 (Context Reachable States)**.** *Let* $L = (E, OT, O, \pi_{act}, \pi_{omap}, \ldots)$ *be an object-centric event log,* $OCPN_A$ *be an accepting object-centric Petri net and* $e \in E$ *be an event. We assume the existence of an oracle states*: $(OCPN_A, context_e) \mapsto P(M)$ *that retrieves the reachable states with a binding sequence of* $context_e$.

For a given context one can collect all the states that are reachable from the initial marking by a binding sequence that produces the same context. We illustrate this with the model of Figure 1 and $context_{e_5}$. The binding sequence σ_1 has the same context and results in marking $[(p5, p1); (p6, b1); (p6, b2)]$. There are four more binding sequences: $\sigma_2 = \sigma_1 \cdot (\ldots, \{b1\})$, $\sigma_3 = \sigma_1 \cdot (\ldots, \{b2\})$, $\sigma_4 = \sigma_1 \cdot (\ldots, \{b1\}) \cdot (\ldots, \{b2\})$ and $\sigma_5 = \sigma_1 \cdot (\ldots, \{b2\}) \cdot (\ldots, \{b1\})$.These are binding sequences that also execute the silent transition. As this does not influence the context they all have the same context. All these possible binding sequences define the four reachable states of the context: $states(OCPN_1, context_{e_5}) =$

$f[(pl5,p1);(pl6,b1);(pl6,b2)]; \quad [(pl5,p1);(pl8,b1);(pl6,b2)]; [(pl5,p1);(pl6,b1);(pl8,b2)]; [(pl5,p1);(pl8,b1);(pl8,b2)]g$.

The allowed behaviour of the model is specified by the enabled activities of the model in any of the reachable states of a context.

Definition 11 (Enabled Model Activities)**.** *Let* $L = (E; OT; O; {}_{act}; {}_{omap};)$ *be an object-centric event log,* $e2E$ *be an event,* $OCPN_A = (OCPN; M_{init}; M_{final})$ *be an accepting object-centric Petri net with* $OCPN = (N; pt; F_{var})$ *and* $N = (P; T; F; l)$*.* $en_{OCPN}(e) = fl(t)j(t;b)2B \wedge 9M2states(OCPN_A; context_e)9M^{0}2B(Q_{OCPN})M \xrightarrow{(t;b)} M^{0}g$ *describes the enabled activities in the model for the corresponding context of an event.*

Applying this to our running example, we extract the enabled activities for any state in $states(OCPN_1; context_{e_5})$. There are the two enabled activities *Lift off* and *Pick up* , i.e., $en_{OCPN_1}(e_5) = fLift\ off; Pick\ up\ @\ destg$.

With the so far introduced concepts we can already derive the context of an event and state the possible behavior of the model for this context. To retrieve precision and fitness of the model we now need to specify the behavior that is given by the log. The behavior recorded in the event log is specified by comparing the subsequent activities for the same context.

Definition 12 (Enabled Log Activities)**.** *Let* $L = (E; OT; O; {}_{act}; {}_{omap};)$ *be an object-centric event log and* $e2E$ *be an event.* $en_L(e) = f {}_{act}(e^{0}) \quad j \quad e^{0}2E \wedge context_{e^{0}} = context_e g$ *defines the enabled log activities for the corresponing context of an event.*

We illustrate that using our running example of $context_{e_5}$. There is one other event that has the same context which is e_{14}. The activity that is executed for both events of this context is *Lift off*, i.e., $en_{L_1}(e_5) = fLift\ offg$.

The enabled log and model activities are calculated for the context of each event and the share of behaviour contained in the log and also allowed by the model, the fitness, is calculated. If all the behavior of the log is also allowed in the model it has a fitness of 1, if all replaying ends up in a final marking one could speak of perfect fitness.

Definition 13 (Fitness)**.** *Let* $L = (E; OT; O; {}_{act}; {}_{omap};)$ *be an object-centric event log and* $OCPN_A$ *be an accepting object-centric Petri net. The fitness of* $OCPN_A$ *with respect to* L *is* $fitness(L; OCPN_A) = \frac{1}{jEj} \sum_{e2E} \frac{jen_L(e) \setminus en_{OCPN_A}(e)j}{jen_L(e)j}$.

The enabled log and model activities are calculated for the context of each event and the share of behaviour allowed by the model and also contained in the log, the precision, is calculated. Not replayable events are skipped. If all the behavior allowed by the model is also contained in the log the model is perfectly precise.

Definition 14 (Precision)**.** *Let* $L = (E; OT; O; {}_{act}; {}_{omap};)$ *be an object-centric event log and* $OCPN_A$ *be an accepting object-centric Petri net.* $E_f = fe^{0}2E \ j \ en_{OCPN_A}(e^{0}) 6= ;g$ *is the set of replayable events. The precision of* $OCPN_A$ *with respect to* L *is calculated by* $precision(L; OCPN_A) = \frac{1}{jE_f j} \sum_{e2E_f} \frac{jen_L(e) \setminus en_{OCPN_A}(e)j}{jen_{OCPN_A}(e)j}$.

Algorithm 1: Enabled model activities of a context

```
Input Event log and context;
Output Enabled model activities of this context;
collect all events that have this context;
for each event with this context do
    extract the binding sequence of visible transitions
      from the event preset;
    create initial state and append to queue;
    while state in queue do
        pop first state of the queue;
        if binding sequence is fully replayed in this
          state then
            Add enabled activities of this state to the
              results;
        if next binding enabled then
            execute binding and enqueue the resulting
              state;
        else
            enqueue all resulting states from enabled
              bindings of silent transitions;
```

The fitness and precision metrics retrieve single comprehensive numbers about the quality of the model. We apply this to our running example. For all the events the enabled activities of the log are also allowed by the model. The fitness of the model is, therefore, $fitness(L_1; OCPN_1) = 1$. The only events where the enabled model activities exceed the enabled log activities are events $e_5; e_6; e_{14}; e_{15}$. For each of these events, *Pick up @ dest* is enabled in the model but not in the log, i.e., the model allows for baggage to be picked up before the baggage was unloaded which is not contained in the event log. We, therefore, retrieve a precision of $precision(L_1; OCPN_1) = 0{:}89$.

V. Calculating Precision and Fitness

In this section, we discuss an algorithm to calculate precision and fitness. Our algorithm is based on replaying the events on the model. The implementation of our algorithm is available on GitHub[1]. The calculation of fitness and precision can be divided in two steps: determining the enabled log activities and the enabled model activities.

Constructing the contexts and calculating the enabled log activities is straightforward according to Definition 8 and Definition 12. We construct the event-object graph for the event log, extract the event preset for each event, calculate the prefix for each object and merge these prefixes into the context. We, subsequently, collect the activities of all events with the same context as the enabled activities of this context.

Due to variable arcs and silent transitions the calculation of enabled model activities is not trivial. We, therefore, introduce

[1] https://github.com/niklasadams/PrecisionFitnessOCPM

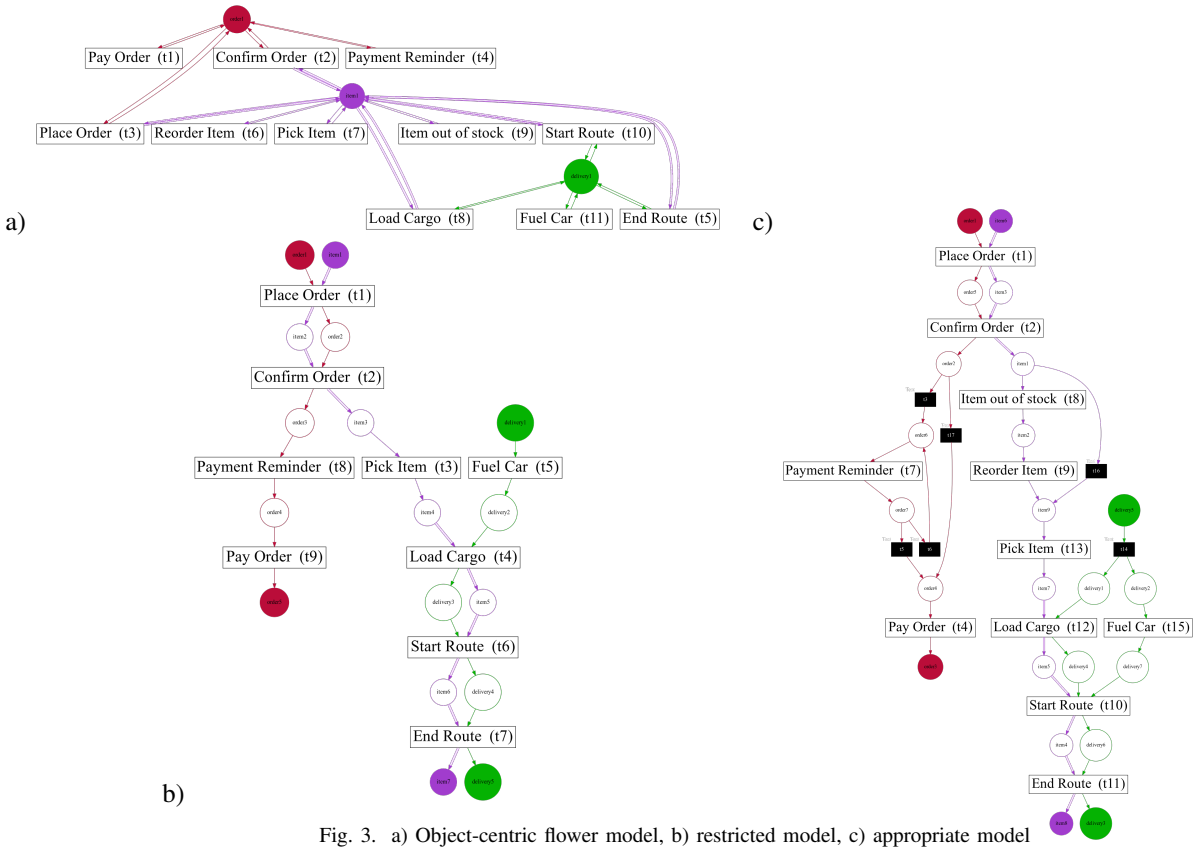

Fig. 3. a) Object-centric flower model, b) restricted model, c) appropriate model

an algorithm to determine the enabled model activities which is depicted in Algorithm 1. The core idea of our algorithm is to replay the events for each context and determine the enabled activities in the resulting states of the object-centric Petri net. For a given context we collect all the events that have this context. For each of these events we extract the binding sequence of visible transitions recorded in the event log. We do this by projecting the event preset of an event onto the bindings described by the events, i.e., transition and objects. See the event log L_1 in Table I and event e_5 with event preset $e_5 = \{e_1, e_2, e_3, e_4\}$ as example. The corresponding binding sequence of visible transitions that need to be replayed are the bindings of e_1 to e_4, consisting of the transition and recorded objects, in order of their occurrence. We replay each binding sequence under consideration of silent transitions. This is a breadth-first search through the states reachable by executing the binding sequence and silent transitions. When the binding sequence is fully replayed all enabled activities are added to the set of enabled model activities for this context.

One important aspect for the practicality of these fitness and precision notion is the complexity of the algorithm. The computation of the measures consists of computing enabled log and enabled model activities. Determining enabled log activities is done in quadratic time since the event log has to be traversed once for each event to determine the preset. The computation of enabled model activities depends on the log and the Petri net. It depends linearly on the number of events, however, two factors can lead to an exponential increase in the computation time: silent transitions and the number of objects. Coherent clusters of silent transitions with choices within one object type lead to the necessity to replay through all the possible reachable states to align the process model with the log, if possible. Especially when considering that multiple objects of one object type can be involved the state space that needs to be searched grows exponentially. This is a problem when considering, e.g., object-centric Petri nets mined by the inductive miner on a log with, e.g., noise.

VI. Evaluation

In this section, we discuss our precision and fitness notion by applying it to three different models of one synthetic event log and analyzing the results. We do this to assess whether our precision and fitness notion can be interpreted analogously to the notions of traditional process mining, i.e., provide an intuitive interpretation for experts and practitioners. We use an object-centric flower model, a model that is tailored to the most frequent process execution and an appropriate model.

The first model is an object-centric adaptation to a flower model, it is depicted in Figure 3a. In traditional process mining, a flower model is used to describe a process where every transition can be executed at any time. The fitness is very high since it covers any behavior seen in the log. The precision, however, is very low since all transitions can happen at any time, also behavior that is not covered by the event log. For an object-centric flow model, we expect a high fitness and

TABLE II
RESULTS FOR DIFFERENT MODELS

	Fitness	Precision	Skipped Events
Flower Model (Figure 3a)	1	0:25	0%
Restricted Model (Figure 3b)	0:31	0:95	54%
Appropriate Model (Figure 3c)	1	0:57	0%

low precision.

The second model is a model that just accounts for the most frequent activity sequence of each object type, it is depicted in Figure 3b. This is a very restrictive model that only allows for little behavior. In traditional process mining, the precision of such a model is high as the little behavior it covers is contained in the log. The fitness, however, would be low since such a model only suits the most frequent execution of the process. We expect a low fitness and high precision for our notions.

The third model is an appropriate representation of the underlying process, it is depicted in Figure 3c. We, therefore, expect high measures for fitness and precision.

The results calculated by our algorithm are displayed in Table II. The appropriate model fits the event log perfectly and has a precision of 0.57[2]. This is a high precision compared to the flower model which also fits perfectly but has a low precision since it always allows for every activity to be executed. The restricted model shows an almost perfect precision. However, missing concurrency, choice and activities lead to a very low fitness and approximately 54% of the events where the context can not be replayed and which can not be considered. In summary, our fitness and precision notion behave analogously to the ones of traditional process mining and, therefore, yield an intuitive and easy-to-understand object-centric definition of fitness and precision.

VII. CONCLUSION

In this paper, we introduced a precision and fitness notion for object-centric Petri nets with respect to an object-centric event log. We use the concept of a context to relate log and model behavior and calculate the enabled activities for the context in the model and the log. We handle contexts that are not replayable on the Petri net by excluding them from the precision calculation. We provide an algorithm and implementation of calculating these quality metrics. We evaluate our contributions by comparing the quality measures for one event log and three different models. Our fitness and precision notion offer an objective way to evaluate the quality of an object-centric Petri net with respect to an object-centric event log. In the future, we want to use these concepts for evaluation of improved object-centric process discovery. Other future lines of research could focus on handling non-replayable context, e.g., by the calculation of object-centric alignments or on providing approximations for models with large state spaces for replay. Furthermore, other quality metrics considered in traditional process mining, i.e., generality and simplicity, could be investigated.

[2]On the first look this precision looks low for an appropriate model. However, consider that the object-centric Petri net makes no limitation on the order of activity executions between object types. If *Pay Order* is always executed before *End Route* this is not represented by the Petri net and, therefore, an imprecision.

REFERENCES

[1] W. M. P. van der Aalst, *Process mining: Data science in action*. Springer, 2016.

[2] ——, "Object-centric process mining: Dealing with divergence and convergence in event data," in *SEFM 2019.*, ser. LNCS, vol. 11724. Springer, 2019, pp. 3–25.

[3] A. F. Ghahfarokhi, G. Park, A. Berti, and W. M. P. van der Aalst, "OCEL: A standard for object-centric event logs," in *ADBIS (Short Papers) 2021*, ser. CCIS, vol. 1450. Springer, pp. 169–175.

[4] W. M. P. van der Aalst and A. Berti, "Discovering object-centric Petri nets," *Fundam. Informaticae*, vol. 175, no. 1-4, pp. 1–40, 2020.

[5] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, no. 1, pp. 64–95, 2008.

[6] J. Munoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *BPM 2010.*, ser. LNCS, vol. 6336. Springer, pp. 211–226.

[7] W. M. P. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer, "Workflow modeling using proclets," in *CoopIS 2000.*, ser. LNCS, vol. 1901. Springer, pp. 198–209.

[8] D. Fahland, "Describing behavior of processes with many-to-many interactions," in *PETRI NETS 2019.*, ser. LNCS, vol. 11522. Springer, pp. 3–24.

[9] V. Denisov, D. Fahland, and W. M. P. van der Aalst, "Multi-dimensional performance analysis and monitoring using integrated performance spectra," in *ICPM 2020*, ser. CEUR, vol. 2703. CEUR-WS.org, pp. 27–30.

[10] D. Cohn and R. Hull, "Business artifacts: A data-centric approach to modeling business operations and processes," *IEEE Data Eng. Bull.*, vol. 32, no. 3, pp. 3–9, 2009.

[11] D. Calvanese, M. Montali, M. Estañol, and E. Teniente, "Verifiable UML artifact-centric business process models," in *CIKM 2014.* ACM, pp. 1289–1298.

[12] X. Lu, M. Nagelkerke, D. van de Wiel, and D. Fahland, "Discovering interacting artifacts from ERP systems," *IEEE Trans. Serv. Comput.*, vol. 8, no. 6, pp. 861–873, 2015.

[13] D. Fahland, M. de Leoni, B. F. van Dongen, and W. M. P. van der Aalst, "Behavioral conformance of artifact-centric process models," in *BIS 2011*, ser. LNBIP, vol. 87. Springer, pp. 37–49.

[14] ——, "Conformance checking of interacting processes with overlapping instances," in *BPM 2011.*, ser. LNCS, vol. 6896. Springer, pp. 345–361.

[15] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst, "Guided interaction exploration in artifact-centric process models," in *CBI 2017*. IEEE Computer Society, pp. 109–118.

[16] M. Montali and A. Rivkin, "Db-nets: On the marriage of colored petri nets and relational databases," *Trans. Petri Nets Other Model. Concurr.*, vol. 12, pp. 91–118, 2017.

[17] W. M. P. van der Aalst, G. Li, and M. Montali, "Object-centric behavioral constraints," *CoRR*, vol. abs/1703.05740, 2017.

[18] G. Li, R. M. de Carvalho, and W. M. P. van der Aalst, "Automatic discovery of object-centric behavioral constraint models," in *BIS 2017.*, ser. LNBIP, vol. 288. Springer, pp. 43–58.

[19] A. Berti and W. M. P. van der Aalst, "Extracting multiple viewpoint models from relational databases," in *SIMPDA Revised Selected Papers*, ser. LNBIP, vol. 379. Springer, 2019, pp. 24–51.

[20] A. F. Syring, N. Tax, and W. M. P. van der Aalst, "Evaluating conformance measures in process mining using conformance propositions," *Trans. Petri Nets Other Model. Concurr.*, vol. 14, pp. 192–221, 2019.

[21] N. Tax, X. Lu, N. Sidorova, D. Fahland, and W. M. P. van der Aalst, "The imprecisions of precision measures in process mining," *Inf. Process. Lett.*, vol. 135, pp. 1–8, 2018.

[22] W. M. P. van der Aalst, A. Adriansyah, and B. F. van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 182–192, 2012.