OPerA: Object-Centric Performance Analysis*

Gyunam Park[0000−0001−9394−6513],
Jan Niklas Adams[0000−0001−8954−4925], and Wil. M. P. van der Aalst[0000−0002−0955−6940]

Process and Data Science Group (PADS), RWTH Aachen University
{gnpark,niklas.adams,wvdaalst}@pads.rwth-aachen.de

Abstract. Performance analysis in process mining aims to provide insights on the performance of a business process by using a process model as a formal representation of the process. Existing techniques for performance analysis assume that a single case notion exists in a business process (e.g., a patient in healthcare process). However, in reality, different objects might interact (e.g., order, delivery, and invoice in an O2C process). In such a setting, traditional techniques may yield misleading or even incorrect insights on performance metrics such as waiting time. More importantly, by considering the interaction between objects, we can define object-centric performance metrics such as synchronization time, pooling time, and lagging time. In this work, we propose a novel approach to performance analysis considering multiple case notions by using object-centric Petri nets as formal representations of business processes. The proposed approach correctly computes existing performance metrics, while supporting the derivation of newly-introduced object-centric performance metrics. We have implemented the approach as a web application and conducted a case study based on a real-life loan application process.

Keywords: Performance Analysis · Object-Centric Process Mining · Object-Centric Petri Net · Actionable Insights · Process Improvement

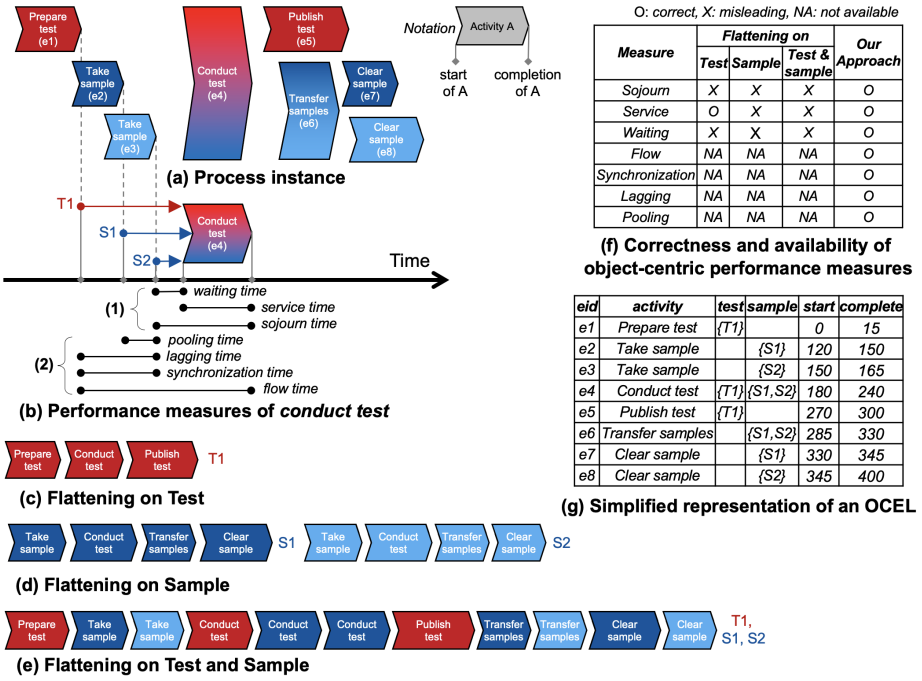
1 Introduction

Process mining provides techniques to extract insights from event data recorded by information systems, including process discovery, conformance checking, and performance analysis [1]. Especially performance analysis provides techniques to analyze the performance of a business process using process models as representations of the process [6].

Existing techniques for performance analysis have been developed, assuming that a single case notion exists in business processes, e.g., a patient in a healthcare process [5, 6, 8, 11–14]. Such a case notion correlates events of a process instance and represents them as a single sequence, e.g., a sequence of events of a patient. However, in real-life business processes supported by ERP systems such as SAP and Oracle, multiple objects (i.e., multiple sequences of events) exist in a process instance [3, 7] and they share events (i.e., sequences are overlapping). Fig. 1(a) shows a process instance in a simple blood test process as multiple overlapping sequences. The red sequence represents the event sequence of test *T1*, whereas the blue sequences indicate the event sequences of samples *S1*

* This work is supported by the Alexander von Humboldt (AvH) Stiftung.
An extended version is available online: https://arxiv.org/abs/2204.10662

and *S2*, respectively. The objects share *conduct test* event (*e4*), i.e., all the sequences overlap, and the samples share *transfer samples* event (*e6*), i.e., the sample sequences overlap.



O: correct, X: misleading, NA: not available

Measure	Flattening on			Our Approach
	Test	Sample	Test & sample	
Sojourn	X	X	X	O
Service	O	X	X	O
Waiting	X	X	X	O
Flow	NA	NA	NA	O
Synchronization	NA	NA	NA	O
Lagging	NA	NA	NA	O
Pooling	NA	NA	NA	O

(f) Correctness and availability of object-centric performance measures

eid	activity	test	sample	start	complete
e1	Prepare test	{T1}		0	15
e2	Take sample		{S1}	120	150
e3	Take sample		{S2}	150	165
e4	Conduct test	{T1}	{S1,S2}	180	240
e5	Publish test	{T1}		270	300
e6	Transfer samples		{S1,S2}	285	330
e7	Clear sample		{S1}	330	345
e8	Clear sample		{S2}	345	400

(g) Simplified representation of an OCEL

Fig. 1: A motivating example showing misleading insights from existing approaches to performance analysis and the proposed object-centric performance analysis

The goal of object-centric performance analysis is to analyze performance in such "object-centric" processes with multiple overlapping sequences using 1) existing performance measures and 2) new performance measures considering the interaction between objects. Fig. 1(b)(1) visualizes existing performance measures related to event *conduct test*. *Waiting time* of *conduct test* is the time spent before conducting the test after preparing test *T1* and samples *S1* and *S2*, while the *service time* is the time spent for conducting the test and *sojourn time* is the sum of *waiting time* and *service time*. Furthermore, Fig. 1(b)(2) shows new performance measures considering the interaction between objects. First, *synchronization time* is the time spent for synchronizing different objects, i.e., samples *S1* and *S2* with test *T1* to conduct the test. Next, *pooling time* is the time spent for pooling all objects of an object type, e.g., the pooling time of *conduct test* w.r.t. *sample* is the time taken to pool the second sample. Third, *lagging* time is the time spent due to the lag of an object type, e.g., the lagging time of *conduct test* w.r.t. *test* is the time taken due to the lag of the second sample. Finally, *flow time* is the sum of *sojourn time* and *synchronization time*.

A natural way to apply existing techniques to multiple overlapping sequences is to *flatten* them into a single sequence. To this end, we select an object type(s) as a case notion, removing events not having the object type and replicating events with multiple objects of the selected type [3]. For instance, Fig. 1(a) is flattened to Fig. 1(c) by using test as a

case notion, to Fig. 1(d) by using sample as a case notion, and Fig. 1(e) by using both test and sample as a case notion.

However, depending on the selection, flattening results in misleading insights. Fig. 1(f) summarizes the correctness of object-centric performance analysis on flattened sequences. 1) Flattening on test provides a misleading waiting time, measured as the time difference between the complete time of *prepare test* and the start time of *conduct test*, and, thus, a misleading sojourn time. 2) Flattening on sample results in misleading insights on the service time since two service times are measured despite the single occurrence of the event. 3) By flattening on both test and sample, the waiting time for *take sample* is measured in relation to *prepare test* although they are independent events from different object types.

In this work, we suggest a novel approach to object-centric performance analysis. The approach uses an Object-Centric Event Log (OCEL) that store multiple overlapping sequences without flattening (cf. Fig. 1(g)) as an input. Moreover, we use Object-Centric Petri Nets (OCPNs) [3] as a formalism to represent process models, and the object-centric performance is analyzed in the context of process models. With formal semantics of OCPNs, we can reliably compute and interpret performance analysis results, considering the concurrency, loops, etc [2].

More in detail, we first discover an OCPN that formally represents a process model from the OCEL. Next, we replay the OCEL on the discovered OCPN to produce *token visits* and *event occurrences*. Finally, we compute object-centric performance measures using the token visit and event occurrence. For instance, in the proposed approach, the waiting time of *Conduct test* is computed as the difference between *e4*'s start and *e1*'s complete. The synchronization time is computed as the time difference between *e3*'s complete and *e1*'s complete.

In summary, we provide the following contributions.

- Our approach correctly calculates existing performance measures in an object-centric setting.
- Our approach supports novel object-centric performance metrics taking the interaction between objects into account, such as synchronization time.
- The proposed approach has been implemented as a web application and a case study with a real-life event log has been conducted to evaluate the effectiveness of the approach.

2 Related Work

Performance analysis has been widely studied in the context of process mining. Table 1 compares existing work and our proposed work in different criteria: 1) if formal semantics exist to analyze performance in the context of process models, 2) if aggregated measures, e.g., mean and median, are supported, 3) if frequency analysis is covered, 4) if time analysis is covered, and 5) if multiple case notions are allowed to consider the interactions of different objects. Existing algorithms/techniques assume a single case notion, not considering the interaction among different objects.

Traditionally, methods in process mining have the assumption that each event is associated with exactly one case, viewing the event log as a set of isolated event sequences.

A demo video and manuals: `https://github.com/gyunamister/OPerA`

Table 1: Comparison of algorithms/techniques for performance analysis

Author	Technique	Form.	Agg.	Freq.	Perf.	Obj.
Maté et al. [13]	*Business Strategy Model*	-	✓	✓	✓	-
Denisov et al. [8]	*Performance Spectrum*	-	✓	✓	✓	-
Hornix [11]	*Petri Nets*	✓	✓	✓	✓	-
Rogge-Solti et al. [14]	*Stochastic Petri Nets*	✓	✓	-	✓	-
Leemans et al. [12]	*Directly Follows Model*	✓	✓	✓	✓	-
Adriansyah et al. [6]	*Robust Performance*	✓	✓	✓	✓	-
Adriansyah [5]	*Alignments*	✓	✓	✓	✓	-
Our work	***Object-Centric***	✓	✓	✓	✓	✓

Object-centric process mining breaks with this assumption, allowing one event to be associated with multiple cases and, thus, having shared events between event sequences. An event log format has been proposed to store object-centric event logs [10], as well as a discovery technique for OCPNs [3] and a conformance checking technique to determine precision and fitness of the net [4]. Furthermore, Esser and Fahland [9] propose a graph database as a storage format for object-centric event data, enabling a user to use queries to calculate different statistics. A study on performance analysis is, so far, missing in the literature, with only limited metrics being supported in [3] by flattening event logs and replaying them. However, object-centric performance metrics are needed to accurately assess performance in processes where multiple case notions occur.

3 Background

Definition 1 (Universes). *Let $\mathbb{U}_{ei}$ be the universe of event identifiers, $\mathbb{U}_{act}$ the universe of activity names, $\mathbb{U}_{time}$ the universe of timestamps, $\mathbb{U}_{ot}$ the universe of object types, and $\mathbb{U}_{oi}$ the universe of object identifiers. $type \in \mathbb{U}_{oi} \to \mathbb{U}_{ot}$ assigns precisely one type to each object identifier. $\mathbb{U}_{omap} = \{omap \in \mathbb{U}_{ot} \not\to \mathscr{P}(\mathbb{U}_{oi}) \mid \forall_{ot \in dom(omap)} \forall_{oi \in omap(ot)} type(oi) = ot\}$ is the universe of all object mappings indicating which object identifiers are included per type. $\mathbb{U}_{event} = \mathbb{U}_{ei} \times \mathbb{U}_{act} \times \mathbb{U}_{time} \times \mathbb{U}_{time} \times \mathbb{U}_{omap}$ is the universe of events.*

Given $e = (ei, act, st, ct, omap) \in \mathbb{U}_{event}$, $\pi_{ei}(e) = ei$, $\pi_{act}(e) = act$, $\pi_{st}(e) = st$, $\pi_{ct}(e) = ct$, and $\pi_{omap}(e) = omap$, where $\pi_{st}(e)$ and $\pi_{ct}(e)$ denotes start and complete timestamps.

Definition 2 (Object-Centric Event Log (OCEL)). *An object-centric event log is a tuple $L = (E, \prec_E)$, where $E \subseteq \mathbb{U}_{event}$ is a set of events and $\prec_E \subseteq E \times E$ is a total order underlying E. $\mathbb{U}_L$ is the set of all possible object-centric event logs.*

Fig. 1(b) describes a fraction of a simple OCEL with two types of objects. For the event in the fourth row, denoted as $e4$, $\pi_{ei}(e4) = e4$, $\pi_{act}(e4) = conduct\ test$, $\pi_{st}(e4) = 180$, $\pi_{ct}(e4) = 240$, $\pi_{omap}(e4)(test) = \{T1\}$, and $\pi_{omap}(e4)(sample) = \{S1, S2\}$. Note that the timestamp in the example is simplified using the relative scale.

Definition 3 (Object-Centric Petri Net (OCPN)). *Let $N = (P, T, F, l)$ be a labeled Petri net with P the set of places, T the set of transitions, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ the flow relation, and $l \in T \not\to \mathbb{U}_{act}$ a labeling function. An* object-centric Petri net *is a tuple $ON = (N, pt, F_{var})$, $pt \in P \to \mathbb{U}_{ot}$ maps places to object types, and $F_{var} \subseteq F$ is the subset of variable arcs.*

Fig. 2(a) depicts an OCPN, $ON_1=(N,pt,F_{var})$ with $N=(P,T,F,l)$ where $P=\{p1,...,p9\}$, $T=\{t1,...,t6\}$, $F=\{(p1,t1),(p2,t2),...\}$, $l(t1)=$*prepare test*, etc., $pt(p1)=test$, $pt(p2)=$ *sample*, etc., and $F_{var}=\{(p4,t3),(t3,p6),...\}$.

A token consists of a place and an object, e.g., $(p3,T1)$ denotes a token of object $T1$ in $p3$. A marking of an OCPN is a multiset of tokens. For instance, marking $M_1=[(p3, T1),(p4,S1),(p4,S2)]$ denotes three tokens, among which place *p3* has one token of object *T1* and *p4* has two tokens of objects *S1* and *S2*.

A binding describes the execution of a transition consuming objects from its input places and producing objects for its output places. A binding (t,b) is a tuple of transition t and function b mapping the object types of the surrounding places to sets of object identifiers. For instance, $(t3,b1)$ describes the execution of transition $t3$ with $b1$ where $b1(test)=\{T1\}$ and $b1(sample)=\{S1,S2\}$, where *test* and *sample* are the object types of its surrounding places (i.e., $p3$, $p4$, $p5$, and $p6$).

A binding (t,b) is *enabled* in marking M if all the objects specified by b exist in the input places of t. For instance, $(t3,b1)$ is enabled in marking M_1 since $T1$, $S1$, and $S2$ exist in its input places, i.e., $p3$ and $p4$. A new marking is reached by *executing* enabled binding (t,b) at M. For instance, as a result of executing $(t1,b1)$, $T1$ is removed from $p3$ and added to $p5$. Besides, $S1$ and $S2$ are removed from $p4$ and added to $p6$, resulting in new marking $M'=[(p5,T1),(p6,S1),(p6,S2)]$.

4 Object-Centric Performance Analysis

This section introduces an approach to object-centric performance analysis. In the approach, we first discover an OCPN based on an OCEL. Next, we replay the OCEL with timestamps on the discovered OCPN to connect events in the OCEL to the elements of OCPN and compute *event occurrences* and *token visits*. Finally, we measure various object-centric performance metrics based on the event occurrence and token visit. The discovery follows the general approach presented in [3]. In the following subsections, we focus on explaining the rest.

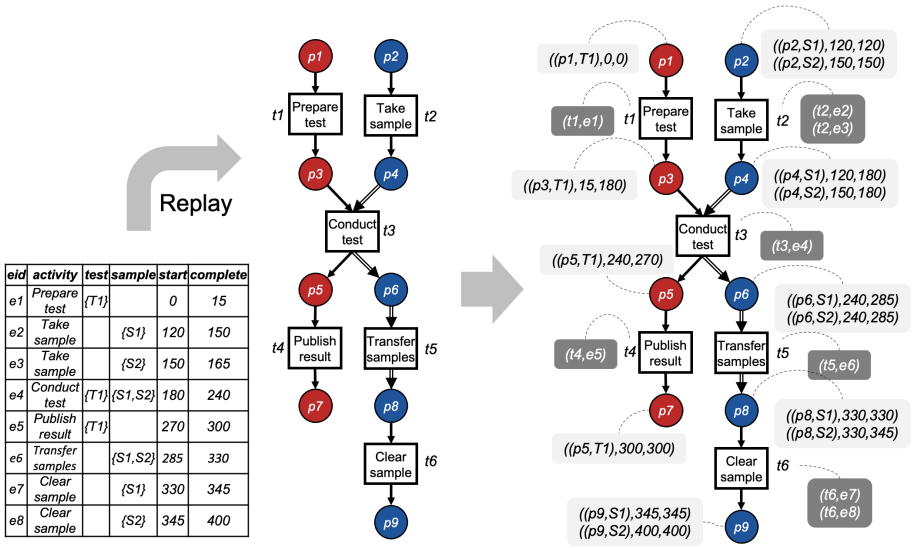
4.1 Replaying OCELs on OCPNs

We couple events in an OCEL to an OCPN by "playing the token game" using the formal semantics of OCPNs. As a result, a set of *event occurrences* are annotated to each visible transition, and a set of *token visits* are recorded for each place. First, an event occurrence represents the occurrence of an event in relation to a transition.

Definition 4 (Event Occurrence). *Let $ON=(N,pt,F_{var})$ be an object-centric Petri net, where $N=(P,T,F,l)$. An event occurrence $eo \in T \times \mathbb{U}_{event}$ is a tuple of a transition and an event. O_{ON} is the set of possible event occurrences of ON.*

For instance, $(t3,e4) \in O_{ON_1}$ indicates that transition *t3* of ON_1 shown in Fig. 2(a) is associated with event *e4*.

A token visit describes "visit" of a token to the corresponding place with the begin time of the visit, i.e., the timestamp when the token is produced, and the end time of the visit, i.e., the timestamp when the token is consumed.



eid	activity	test	sample	start	complete
e1	Prepare test	{T1}		0	15
e2	Take sample		{S1}	120	150
e3	Take sample		{S2}	150	165
e4	Conduct test	{T1}	{S1,S2}	180	240
e5	Publish result	{T1}		270	300
e6	Transfer samples		{S1,S2}	285	330
e7	Clear sample		{S1}	330	345
e8	Clear sample		{S2}	345	400

(a) Replaying L_1 on object-centric Petri net ON_1

(b) Event occurrence O_1 and token visit V_1

Fig. 2: An example of replaying object-centric event logs on an object-centric Petri net

Definition 5 (Token Visit). *Let $ON=(N,pt,F_{var})$ be an object-centric Petri net, where $N=(P,T,F,l)$. $Q_{ON}=\{(p,oi)\in P\times\mathbb{U}_{oi}\mid type(oi)=pt(p)\}$ is the set of possible tokens. A token visit $tv\in Q_{ON}\times\mathbb{U}_{time}\times\mathbb{U}_{time}$ is a tuple of a token, a begin time, and an end time. V_{ON} is the set of possible token visits of ON.*

Given token visit $tv=((p,oi),bt,et)$, $\pi_p(tv)=p$, $\pi_{oi}(tv)=oi$, $\pi_{bt}(tv)=bt$, and $\pi_{et}(tv)=et$. For instance, $((p3,T1),15,180)\in V_{ON_1}$ represents that token $(p3,T1)\in Q_{ON_1}$ is produced in place *p3* at 15 and consumed at 180.

Given an OCEL, a replay function produces event occurrences and token visits of an OCPN, connecting events in the log to the model.

Definition 6 (Replay). *Let ON be an object-centric Petri net. A replay function $replay_{ON}\in\mathbb{U}_L\rightarrow\mathscr{P}(O_{ON})\times\mathscr{P}(V_{ON})$ maps an event log to a set of event occurrences and a set of token visits.*

Fig. 2(b) shows the result of replaying the events in L_1 shown in Fig. 2(a) on model ON_1 depicted in Fig. 2(a). The dark gray boxes represent event occurrences O_1 and the light gray boxes represent token visits V_1, where $replay_{ON_1}(L_1)=(O_1,V_1)$. For instance, replaying event *e1* and *e4* in L_1 produces event occurrences, $(t1,e1)$ and $(t3,e4)$, respectively, and token visit $((p3,T1),15,180)$ where 15 is the time when *e1* completes and 180 is the time when *e4* starts.

4.2 Measuring Object-Centric Performance Measures

We compute object-centric performance measures per event occurrence. For instance, we compute *synchronization*, *pooling*, *lagging*, and *waiting* time of $(t3,e4)$ that analyzes an

event of *conduct test*. To this end, we first relate an event occurrence to the token visits 1) associated with the event occurrence's transition and 2) involving the objects linked to the event occurrence's event.

Definition 7 (Relating An Event Occurrence to Token Visits). *Let L be an object-centric event log and ON an object-centric Petri net. Let $eo=(t,e) \in O$ be an event occurrence. $OI(eo)=\bigcup_{ot \in dom(\pi_{omap}(e))} \pi_{omap}(e)(ot)$ denotes the set of objects related to the event occurrence. $rel_{ON} \in O_{ON} \times \mathscr{P}(V_{ON}) \rightarrow \mathscr{P}(V_{ON})$ is a function mapping an event occurrence and a set of token visits to the set of the token visits related to the event occurrence, s.t., for any $eo \in O_{ON}$ and $V \subseteq V_{ON}$, $rel_{ON}(eo,V)=\bigcup_{oi \in OI(eo)} argmax_{tv \in \{tv' \in V \mid \pi_p(tv') \in \bullet t \wedge \pi_{oi}(tv')=oi\}} \pi_{bt}(tv)$.*

Fig. 3(a) shows the token visits related to $eo_1=(t3,e4)$. $rel_{ON_1}(eo_1,V_1)=\{tv_1=((p3,T1),15,180), tv_2=((p4,S1),120,180), tv_3=((p4,S2),150,180)\}$ since $p3,p4 \in \bullet t3$, $\{T1,S1,S2\} \subseteq OI(eo_1)$, and each token visit is with the latest begin time among other token visits of the corresponding object, e.g., tv_1 is the only (and thus the latest) token visit of $T1$.

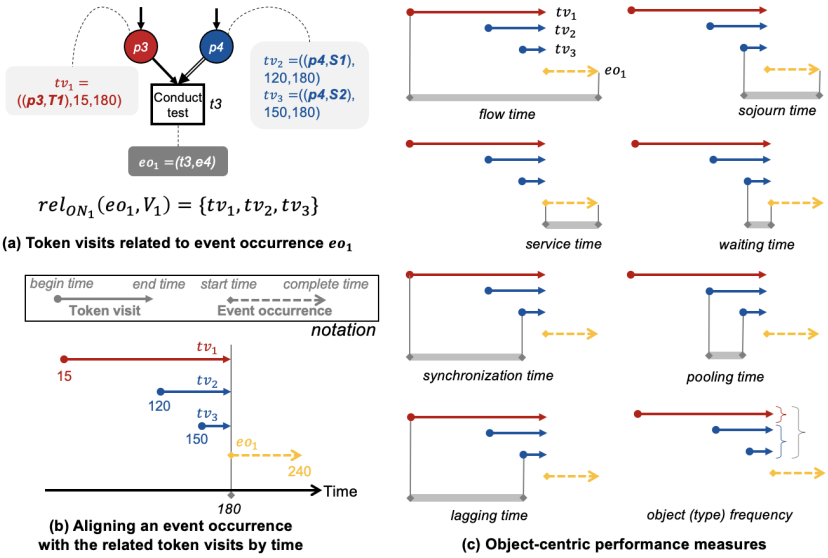


Fig. 3: An example of the token visits related to an event occurrence and object-centric performance measures of the event occurrence.

A measurement function computes a performance measure of an event occurrence by using the related token visits.

Definition 8 (Measurement). *Let ON be an object-centric Petri net. $measure \in O_{ON} \times \mathscr{P}(V_{ON}) \rightarrow \mathbb{R}$ is a function mapping an event occurrence and its related token visits to a performance value. $\mathbb{U}_m$ denotes the set of all such functions.*

In this paper, we introduce seven functions to compute object-centric performance measures as shown in Fig. 3(c). With L an OCEL, ON an OCPN, and $(O,V)=replay_{ON}(L)$, we introduce the functions with formal definitions and examples as below:

- $flow \in \mathbb{U}_m$ computes *flow time*. Formally, for any $eo=(t,e) \in O$, $flow(eo,V)=\pi_{ct}(e)-min(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V)\}$.
- $sojourn \in \mathbb{U}_m$ computes *sojourn time*. Formally, for any $eo=(t,e) \in O$, $sojourn(eo,V)=\pi_{ct}(e)-max(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V)\}$.
- $wait \in \mathbb{U}_m$ computes *waiting time*. Formally, for any $eo=(t,e) \in O$, $wait(eo,V)=\pi_{st}(e)-max(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V)\}$.
- $service \in \mathbb{U}_m$ computes *service time*. Formally, for any $eo=(t,e) \in O$, $service(eo,V)=\pi_{ct}(e)-\pi_{st}(e)$.
- $sync \in \mathbb{U}_m$ computes *synchronization time*. Formally, for any $eo=(t,e) \in O$, $sync(eo,V)=max(T)-min(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V)\}$.
- $pool_{ot} \in \mathbb{U}_m$ computes *pooling time* w.r.t. object type ot. Formally, for any $eo=(t,e) \in O$, $pool_{ot}(eo,V)=max(T)-min(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V) \wedge type(\pi_{oi}(tv))=ot\}$.
- $lag_{ot} \in \mathbb{U}_m$ computes *lagging time* w.r.t. object type ot. Formally, for any $eo=(t,e) \in O$, $lag_{ot}(eo,V)=max(T')-min(T)$ with $T=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V)\}$ and $T'=\{\pi_{bt}(tv) \mid tv \in rel_{ON}(eo,V) \wedge type(\pi_{oi}(tv)) \neq ot\}$ if $max(T') > min(T)$. 0 otherwise.

5 Case Study

The approach discussed in Sec. 4 has been fully implemented as a web application with a dedicated user interface. Using the implementation, we conduct a case study on a real-life loan application process of a Dutch Financial Institute. Two object types exist in the process: *application* and *offer*. An application can have one or more offers. First, a customer creates an application by visiting the bank or using an online system. In the former case, *submit* activity is skipped. After the completion and acceptance of the application, the bank offers loans to the customer by sending the offer to the customer and making a call. An offer is either accepted or canceled.

In this case study, we focus on the offers canceled due to various reasons. We filter infrequent behaviors by selecting the ten most frequent types of process executions. Moreover, we remove redundant activities, e.g., status updates such as *Completed* after *Complete application*. The resulting event log, available at Github repository, contains 20,478 events by 1,682 applications and 3,573 offers.

First, we compare our approach to a traditional technique for performance analysis based on alignments [5]. To apply the traditional technique, we first flatten the log using the application and offer as a case notion. Fig. 4(a) shows the performance analysis results from *Inductive Visual Miner* in *ProM* framework. As shown in ①, 1,799 applications repeat activity *Send*. In reality, as shown in 1, no repetition occurs while the activity is conducted once for each offer except 92 offers skipping it. Furthermore, the average sojourn time for the activity is computed as around 2 days and 23 hours, whereas, in reality, it is around 15 minutes as shown in 1.

Furthermore, ② shows that activity *Cancel application* is repeated 1891 times, but it occurs, in reality, 1,682 times for each application, as depicted in 2. In addition, the average sojourn time for the activity is measured as around 12 days and 22 hours, but in fact, it is around 31 days and 22 hours, as shown in 2.

doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

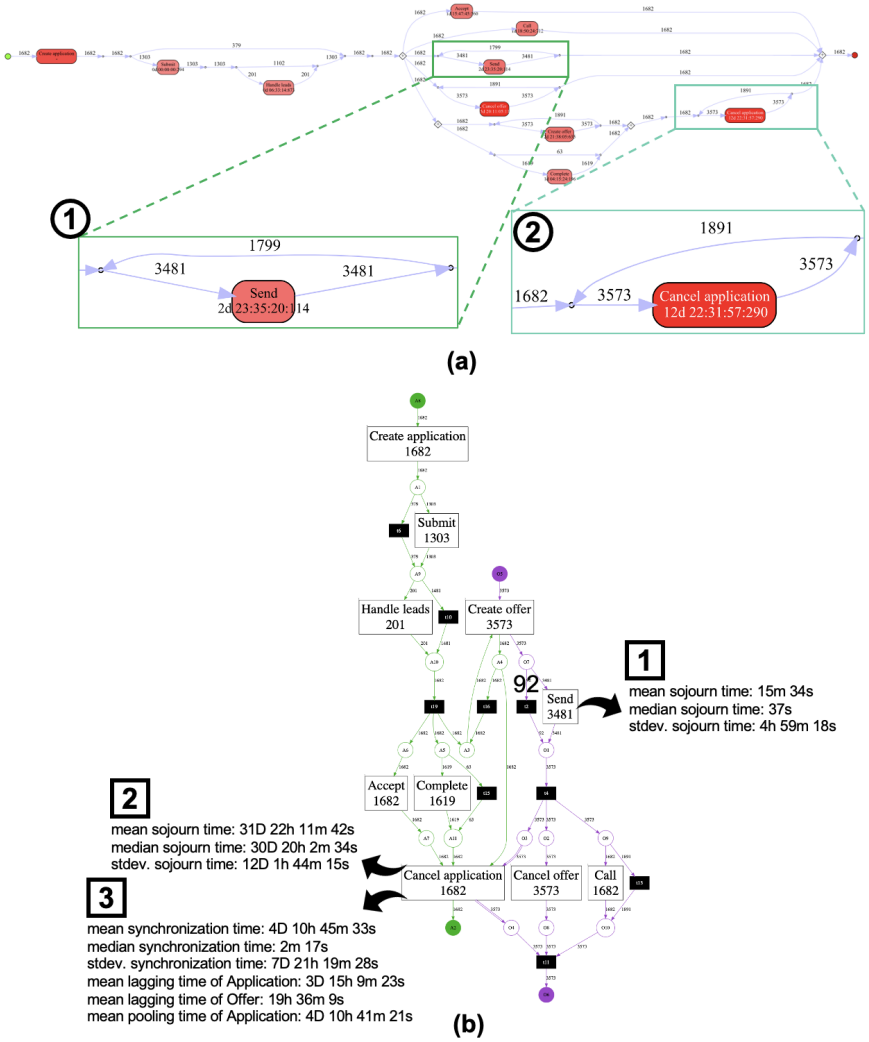


Fig. 4: (a) Performance analysis results based on *Inductive Visual Miner* in *ProM* framework and (b) Performance analysis results based on our proposed approach.

Next, we analyze the newly-introduced object-centric performance measures, including synchronization, lagging, and pooling time. As described in 3, the average synchronization time of activity *Cancel application* is around 4 days and 11 hours. Moreover, the average lagging time of *applications* is 3 days and 15 hours and the lagging time of *offers* is 19 hours, i.e., *offers* are more severely lagging *applications*. Furthermore, the pooling time of *offers* is almost the same as the synchronization time, indicating that the application is ready to be cancelled almost at the same time as the first offer, and the second offer is ready in around 4 days and 11 hours.

6 Conclusion

In this paper, we proposed an approach to object-centric performance analysis. To that end, we first replay OCELs on OCPNs to couple events to process models, producing event occurrences and token visits. Next, we measure object-centric performance metrics per event occurrence by using the corresponding token visits of the event occurrence. We have implemented the approach as a web application and conducted a case study using a real-life loan application process.

The proposed approach has several limitations. First, our approach relies on the quality of the discovered process model. Discovering process models that can be easily interpreted and comprehensively reflect the reality is a remaining challenge. Second, non-conforming behavior in event data w.r.t. a process model can lead to misleading insights. As future work, we plan to extend the approach to support reliable performance analysis of non-conforming event logs. Moreover, we plan to develop an approach to object-centric performance analysis based on event data independently from process models.

References

1. van der Aalst, W.M.P.: Process Mining, Second Edition. Springer (2016)
2. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining Knowl. Discov. **2**(2), 182–192 (2012)
3. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020)
4. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM 2021. pp. 128–135 (2021)
5. Adriansyah, A.: Aligning observed and modeled behavior. Ph.D. thesis, Mathematics and Computer Science (2014)
6. Adriansyah, A., Dongen, van, B., Piessens, D., Wynn, M., Adams, M.: Robust performance analysis on YAWL process models with advanced constructs. Journal of Information Technology Theory and Application **12**(3), 5–26 (2011)
7. Bayomie, D., Ciccio, C.D., Rosa, M.L., Mendling, J.: A probabilistic approach to event-case correlation for process mining. In: ER 2019. pp. 136–152 (2019)
8. Denisov, V., Fahland, D., van der Aalst, W.M.P.: Unbiased, fine-grained description of processes performance from event data. In: BPM 2018, pp. 139–157 (2018)
9. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**(1-2), 109–141 (2021)
10. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL. In: Bellatreche, L., et. al. (eds.) SIMPDA 2021. pp. 169–175 (2021)
11. Hornix, P.T.: Performance analysis of business processes through process mining. Master's thesis, Mathematics and Computer Science (2007)
12. Leemans, S.J.J., Poppe, E., Wynn, M.T.: Directly follows-based process mining: Exploration & a case study. In: ICPM 2019. pp. 25–32 (2019)
13. Maté, A., Trujillo, J., Mylopoulos, J.: Conceptualizing and specifying key performance indicators in business strategy models. In: ER 2012. pp. 282–291 (2012)
14. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: ICSOC. pp. 389–403 (2013)