OCπ: Object-Centric Process Insights

Jan Niklas Adams[1][0000−0001−8954−4925] and Wil M.P. van der Aalst[1,2][0000−0002−0955−6940]

[1] Process and Data Science, RWTH Aachen University, Aachen, Germany
{niklas.adams, wvdaalst}@pads.rwth-aachen.de
[2] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany

Abstract. Process mining uses event sequences recorded in information systems to discover and analyze the process models that generated them. Traditional process mining techniques make two assumptions that often do not find correspondence in real-life event data: First, each event sequence is assumed to be of the same type, i.e., all sequences describe an instantiation of the same process. Second, events are assumed to exclusively belong to one sequence, i.e., not being shared between different sequences. In reality, these assumptions often do not hold. Events may be shared between multiple event sequences identified by objects, and these objects may be of different types describing different subprocesses. Assuming "unshared" events and homogeneously typed objects leads to misleading insights and neglects the opportunity of discovering insights about the interplay between different objects and object types. Object-centric process mining is the term for techniques addressing this more general problem setting of deriving process insights for event data with multiple objects. In this paper, we introduce the tool OCπ. OCπ aims to make the process behind object-centric event data transparent to the user. It does so in two ways: First, we show frequent process executions, defined and visualized as a set of event sequences of different types that share events. The frequency is determined with respect to the activity attribute, i.e., these are object-centric variants. Second, we allow the user to filter infrequent executions and activities, discovering a mainstream process model in the form of an object-centric Petri net. Our tool is freely available for download[1].

Keywords: Process Mining · Object-Centric Petri Net · Process Discovery · Object-Centric Variants.

1 Introduction

Process mining is an umbrella term describing techniques to derive data-driven insights into processes. The data come in an event log, describing the event sequences of many process executions and their associated data. Typically, three different process mining fields are considered: process discovery, conformance checking, and process enhancement [1]. Techniques from process discovery aim to

[1] http://ocpi.ai/

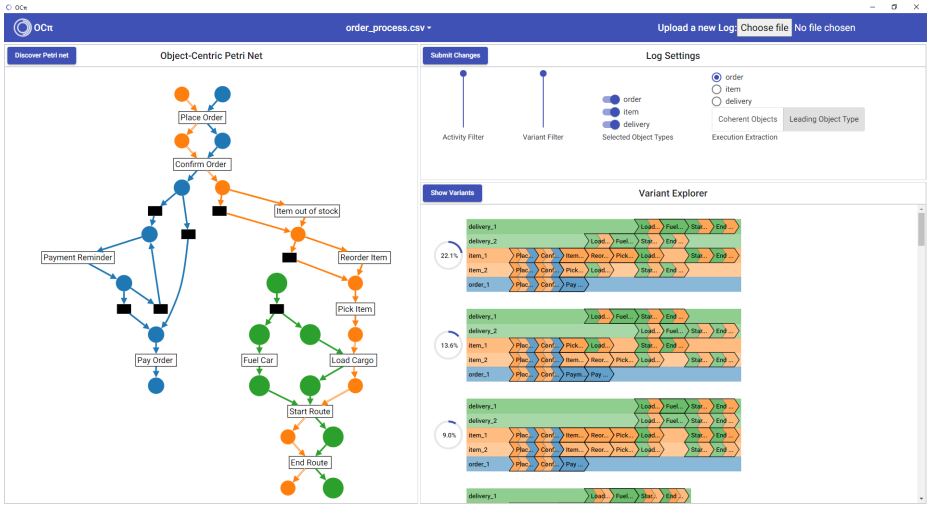

Fig. 1. Overview of OCπ: The log management is integrated into the toolbar, the user can further interact with the tool by submitting the desired settings through the log settings component. Petri net and object-centric variants can be explored through scrolling, zooming and panning.

construct a model from the event log, e.g., a Petri net or a BPMN model. Such a model aims to describe the event sequences contained in the event log with only one comprehensive model. Conformance checking deals with quantifying and describing how well a model corresponds to an event log. Process enhancement aims to deliver data-driven process improvements.

Process mining techniques make some assumption about the nature of event logs and the event sequences event logs contain. Most techniques make the following two assumptions: **(1)** An event log contains process executions related to individual objects, often called cases. Therefore, every event sequence describes an execution of a single case. Each case is of the same case notion. **(2)** Event sequences for cases are independent of each other, i.e., two cases do not share events.

In reality, event logs often violate these assumptions. There is often no clear case notion in an event log. Events can be associated with multiple cases [2]. Imagine an ordering process: An order of some items is placed in a system. These items are later delivered. Events can refer to a case notion of an order, an item, or a delivery. Furthermore, some events are shared between different cases of different case notions, e.g., an event that describes the placing of an order of two items.

In traditional process mining, we cannot derive insights from such event data as a whole. One case notion would need to be chosen, and events referring to multiple objects of that notion would need to be duplicated, called *flattening* [2]. This procedure removes essential information about interactions between different case notions and objects. These problems are the motivation for *object-*

centric process mining [3]. By dropping the two mentioned assumptions and adapting process mining techniques, object-centric process mining aims to deliver algorithms that are able to exploit event logs with multiple case notions and shared events fully.

Different methods for dealing with multiple case notion processes exist. Some, like artifacts [7,8] and proclets [11], deal with the problem mostly from a modeling perspective. Object-centric process mining [2] takes the object-centric event data [12] as a starting point to discover process models and insights. So far, the discovery of process models in the form of object-centric Petri nets [3] has been introduced. A discussion of sound object-centric workflow nets has recently been published [14]. Furthermore, basic conformance checking techniques for object-centric Petri nets and event logs [4], and performance analysis measures [16] have been introduced. On the tool side, tools to extract object-centric Petri nets [3] and object-centric directly-follows graphs [5] as well es storing and querying multiple case notion event data in the form of graph databases [10] have been introduced. Furthermore, object-centric Petri nets have been used to model digital twins of organizations [15].

However, some key ingredients of traditional process mining are, so far, not available to users. Here, we focus on variant visualization and discovering process models for frequent variants. Process executions can be equivalent if they describe the same execution sequences of event activities. The equivalence classes they form are commonly known as *variants* [9]. Each variant has a frequency determined by the number of process executions in this variant. Filtering and exploring frequent variants provides the user with insights into the mainstream behavior of the underlying process and yields a mainstream model.

Therefore, with OCπ (cf. Figure 1), we provide a tool that augments object-centric process discovery in the following two ways:

1. We allow the user to filter the least frequent variants of process executions from the retrieved event log to discover an object-centric Petri net that shows the mainstream behavior of the event log.
2. We provide a variant explorer that allows the user to retrieve and explore the variants of process executions and their frequencies.

The remainder of this paper is structured as follows. In Section 2, we introduce some basic concepts on which this tool is built, i.e., object-centric event logs, extraction of executions, variants, and object-centric Petri nets. We explain the algorithmic foundations and some concepts relevant for the understanding and usage of the tool in Section 3. We provide an extensive overview of the functionalities, the implementation and installation requirements in Section 4. We conclude this tool paper in Section 5.

2 Object-Centric Process Mining

In this section, we introduce some of the basic concepts on which this tool is built. The tool takes input in the form of an object-centric event log, extracts

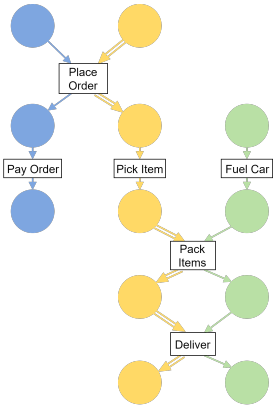

Fig. 2. Example of an object-centric Petri net. Places are colored according to the object type they belong to; variable arcs (double lined) can consume a variable amount of tokens.

Event	Activity	Order	Item	Delivery
e_1	Place Order	o1	i1,i2	
e_2	Pick item		i2	
e_3	Pick item		i1	
e_4	Fuel car			d1
e_5	Pay order	o1		
e_6	Pack items		i1,i2	d1
e_7	Deliver		i1,i2	d1

Table 1. Example of an object-centric event log. Each event can be associated to multiple objects of different object types. In this log, order, item and delivery are the object types.

process executions from it, determines frequent equivalence classes, i.e., variants, and discovers and displays an object-centric Petri net to the user. Therefore, we give a short formal introduction of these concepts in this section.

An object-centric event log can be seen as an extension to traditional event logs used in process mining [1] that records multiple case notions (object types) for each event and allows referencing to multiple cases (objects) of each object type of an event.

Definition 1 (Object-Centric Event Log). *Let U_E be the universe of event identifiers, U_{OT} be the universe of object types, U_O be the universe of objects and U_A be the universe of activities. $\mathcal{P}(X)$ denotes the power set of a set X. $\pi_{ot} : U_O \to U_{OT}$ maps an object to its object type. An object-centric event log is a tuple $L = (E, OT, O, A, \pi_o, \pi_a, \prec)$ consisting of event identifiers $E \subseteq U_E$, object types $OT \subseteq U_{OT}$, objects $O \subseteq U_O$, activities $A \subseteq U_A$, a mapping function from events to objects $\pi_o : E \to \mathcal{P}(O)$ and a mapping function from events to activities $\pi_a : E \to A$. The event identifiers are subject to a total order $\prec$.*

An example of an object-centric event log in table format is given in Table 1. Each event has a unique identifier e_i and an activity[2]. Furthermore, each event has reference to a set of objects. Each object is associated with one object type of *order, item* or *delivery*. In traditional process mining, each process execution is associated with exactly one object, i.e., each process execution is one sequence. The notion of a process execution can be generalized for object-centric event data, involving the sequences multiple objects sharing events. An extraction technique retrieves a set of process executions from an event log.

[2] We omit the timestamp and additional attributes as they are not relevant for the capabilities described in this paper.

Definition 2 (Process Execution). *Let* $L = (E, OT, O, A, \pi_o, \pi_a, \prec)$ *be an object-centric event log.* $P_L = \{(E', O') \mid E' \subseteq E \wedge O' \subseteq O \wedge e \in E' \Leftrightarrow \pi_o(e) \cap O' \neq \emptyset \wedge (O', \{(o, o') \in O' \times O' \mid \exists_{e \in E'} o, o' \in \pi_o(e)\})$ *is a connected graph*$\}$ *is the set of process executions of an event log. An extraction technique* $f_{extract} : L \to \mathcal{P}(P_L)$ *extracts a subset of all process executions.*

For the example of the event log excerpt in Table 1, a single process executions could be all seven events and the four objects. Another possible process execution would be a subset of these objects and their events, e.g., only the order o1 and the two items i1 and i2. Different methods are available to retrieve subsets of all process executions. The process executions in the extracted subset should have some similar characteristics to be comparable. We discuss two different extraction techniques in Subsection 3.2.

Variants in process mining summarize multiple cases (or process executions) with the same control-flow behavior. This is translated to the object-centric setting by determining equivalency of process executions concerning the event activity attribute and grouping equivalent executions in one class, i.e., variant.

Definition 3 (Equivalent Process Executions). *Let* $P = \{p_1, p_2, \ldots p_n\}$ *be a set of process executions. An oracle* $f_{equiv} : P \to \{1, \ldots, m\}$ *maps executions to* $m \in \mathbb{N}$ *classes of equivalent executions considering the event's activity. Each class is one variant* $V_i = \{p_j \in P \mid f_{equiv}(p_j) = i\}$ *for* $i \in \{1, \ldots, m\}$.

We can discover an object-centric Petri net from an object-centric event log [3]. Object-centric Petri nets borrow from colored Petri nets [13] to be able to model different object types and how they interact.

Definition 4 (Object-Centric Petri Net). *Let* $N = (P, T, F, l)$ *be a Petri net with places* P*, transitions* T*, a flow relation* $F \subseteq T \times P \cup P \times T$ *with* $T \cap P = \emptyset$ *and a labelling function* $l : T \nrightarrow U_A$ *and let* $OT \subseteq U_{OT}$ *be a set of object types. An object-centric Petri net* $OCPN = (N, pt, f_{var})$ *is a tuple of a Petri net* N*, a mapping function from places to object types* $pt : P \to OT$ *and* $f_{var} \subseteq F$ *describing a subset of arcs which are variable arcs, i.e., they can consume and produce more than one token.*

An example of an object-centric Petri net is given in Figure 2. This Petri net describes the process used to generate the event log of Table 1. Each place has a color corresponding to one object type. Each arc can either be a standard arc or a variable arc. When playing the token game in such a Petri net, a binding execution of a transition would consume tokens associated with objects in the input places. These could be multiple tokens in the case of a variable arc. The consumed tokens are then produced in the output places of the corresponding object types of the input places.

3 Algorithmic Concept

The technique to discover an object-centric Petri net from an object-centric event log is described in [3]. We focus on preprocessing the underlying event log

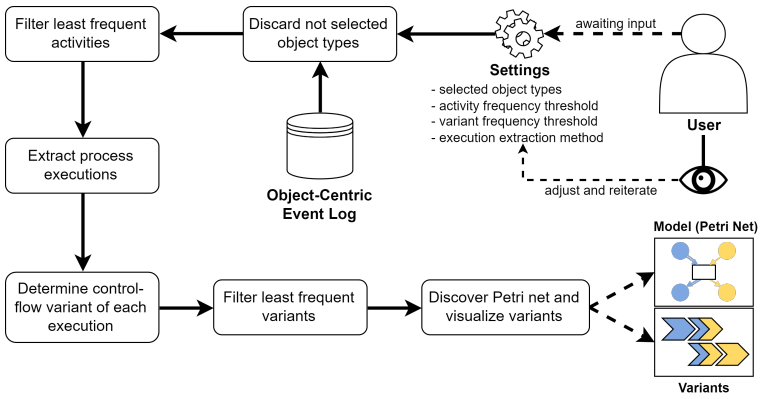

Fig. 3. Overview over the user interaction and the algorithmic steps in OCπ.

for process discovery according to some user input. The general signature of the processing we apply is depicted below.

$$L \xrightarrow{f_{extract}} \{p_1, \ldots, p_n\} \xrightarrow{f_{equiv}} \{V_1, \ldots, V_m\} \rightarrow \text{Petri net \& Variants}$$

The detailed algorithmic concept of our tool is depicted in Figure 3. The user provides an object-centric event log and makes a choice about several settings: The selected object types, the activity threshold, the variant threshold, and the technique to extract process executions. Based on this, our tool applies different filtering and algorithmic steps. Process executions are extracted from the object-centric event log, and their equivalence classes, i.e., variants, are calculated. Subsequently, the user can explore the object-centric Petri net and the variants, adjust the input settings, and reiterate until the result is sufficient for the user. In the following sections, we provide deeper insights into the different algorithmic steps conducted by our tool that are of importance for the understanding of the user on how to interpret the results.

3.1 Filtering

The filtering possibilities included in our tool, by activity frequency and variant frequency, both follow the same method: The user selects a threshold between 0 and 1.0. Subsequently, the minimum number of behavioral observations, i.e., either activities or variants, is collected such that the cumulative frequency of these observations exceeds this threshold. The events that are not associated with these observations are filtered out. For the example of activity filtering, the relative frequencies of each activity in the event log are calculated. After the user sets a threshold, the most frequent activities are greedily added to a set of activities that should be kept in the event log until the threshold is met. All events with activities not in this set will be discarded.

3.2 Process Execution Extraction

The process execution extraction technique determines how process executions are retrieved from the object-centric event log and is, therefore, important for the variants retrieved as well as the results of the variant filtering. We provide two execution extraction techniques: *coherent objects* and *leading object type*.

For a brief explanation of these techniques, we use a concept of direct relations between objects: if two objects share an event, they are directly related. Two objects can be transitively related if a chain of direct relations leads from one object to the other. The length of the chain is the level of transitivity. Coherent objects take all objects directly and transitively related into one process execution. The underlying assumption is that they are all dependent on each other by sharing events. However, this might lead to process executions that are too extensive for some logs and some users. Imagine the ordering process from Table 1. If multiple items of multiple orders end up in the same delivery, all of these orders would be one process execution. While they, indeed, all depend on each other, this might be too extensive for the user who may only be interested in the execution of an order and the associated objects or a delivery and the associated objects. Because of this, we include leading object type as a technique for execution extraction. It constructs executions by taking each object of the leading object type and recursively adding directly related objects until objects of the same type have already been added on a lower level of transitivity. These objects are not added anymore, and their directly related objects are not further traversed. The events of the selected objects are one process execution. This execution extraction cuts off some dependencies to have more understandable and comprehensive process executions. Some events may end up in multiple process executions. Our implementation determines the equivalence of process executions by testing for automorphism between the different execution graphs given by the process executions.

3.3 Variant Visualization

Each execution is associated with one variant, describing the sequence of activities for each involved object. We visualize these variants by giving each object one lane. Each object gets a color, dependent on the object type's base color, which is slightly altered for each object. We draw a chevron for each event. The activity is depicted inside the chevron. If an event is shared between objects, the corresponding chevron is drawn on each lane and colored with all colors of all involved objects. Generally, the chevrons are of the same width. Only if a chevron is placed between two shared chevrons, the width is adjusted to fit the gap. Except for the shared chevrons, the chevron ordering and horizontal positioning gives information about the time-ordering within one object, not between objects.

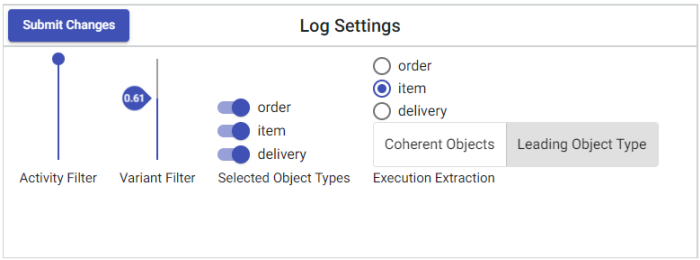

Fig. 4. The log settings component, the central point of interaction with the users.

4 Usage and Functionalities

In this section, we introduce the core functionalities of our tool. A complete view of the tool is depicted in Figure 1. Our tool is separated into four components: two for user interaction and two for exploring the output. The user interaction components are the log management (integrated into the toolbar on top) and the log settings. The output components are the object-centric Petri net explorer and the variant explorer. We are going to introduce each of the components in the following sections.

4.1 Event Log Management

OCπ offers extensive event log management. The file formats *jsonocel* and *jsonxml* introduced in the OCEL standard [12] as well as a *csv* import are supported. The CSV file should contain an "event_activity" and "event_timestamp" column. As a supportive element for uploading CSV, we implemented functionality to choose potential object types to prevent unwanted columns from becoming object types. The file's encoding should comply with UTF-8 encoding. The separator will automatically be detected. Each log can also be deleted at an arbitrary point in time. Each event log will be uniquely identified by its name. Two event logs with the same filename cannot be uploaded.

4.2 Log Settings

The log setting component is the main point of input from the user and is depicted in Figure 4. We provide two filtering thresholds: One to filter out infrequent activities and one to filter out infrequent variants. Furthermore, we allow the user to discard some object types for their scope of analysis, effectively removing these object types and their objects from the event log. We provide two different techniques to extract process executions. The first one is called *coherent objects*, the second technique is called *leading object type*. Subsection 3.2 provides a detailed explanation of the execution extraction. The settings can be submitted to the back-end by pressing the *submit changes* button.

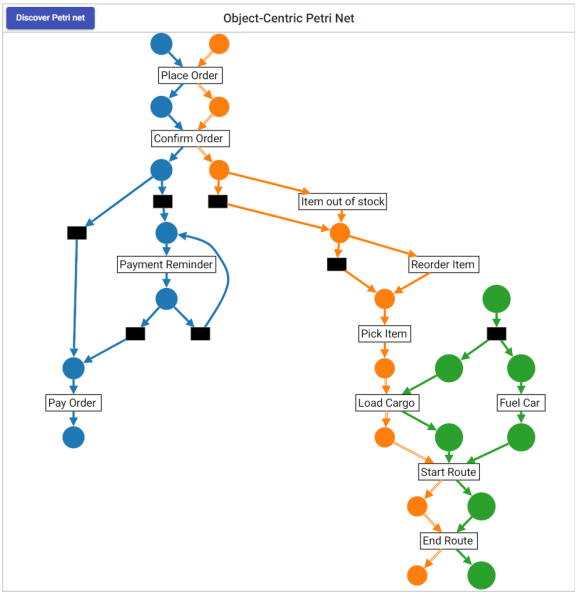

Fig. 5. The object-centric Petri net discovered from an object-centric event log according to the settings provided by the user.

4.3 Process Model Explorer

Based on the event log that is processed according to the setting provided by the user, our tool discovers an object-centric Petri net and displays it to the user as a process model. The component is depicted in Figure 5. The visualization can interactively be explored, navigation by zooming and dragging/panning is supported. These functionalities help make large Petri nets with many object types and transitions accessible to users. Every object type has one globally assigned color, which is also used by the variant explorer component described later. The Petri net can be discovered by pressing the button. If the settings were changed and the Petri net is not consistent with the current submitted settings, this button changes its color to red to indicate a necessary update.

4.4 Variant Explorer

The variant explorer displays the variants of the process executions extracted from the event log based on the provided settings. It is depicted in Figure 6. The colors of the object types are consistent with the colors in the object-centric Petri net component. Each object of an object type is colored in a different shade of the object types' base color. Each object gets a lane to describe its event sequence, lanes of the same object types are grouped. The frequency of the variant (with respect to the event log after the filtering settings are applied) is depicted on the left-hand side of the variant. With a click on the variant, the

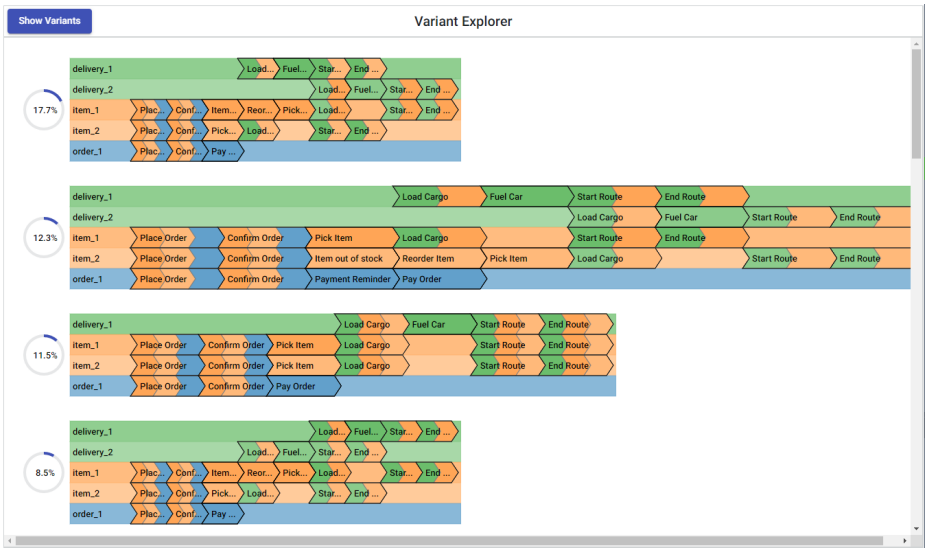

Fig. 6. The variant explorer shows the frequency and a visualisation for each variant.

variant is unfolded such that the full activity labels are visible. The explorer allows scrolling in both directions, vertically and horizontally.

4.5 Implementation

We implemented this tool on a technology stack of Python, Django, Angular, and D3.js and GraphViz[3] for visualization. The core algorithmic functionality is taken from the OCPA[4] and the PM4Py library [6]. The tool can be run on Windows by downloading it from `http://ocpi.ai/` and running the executable named *OCpi.exe*.

5 Conclusion

In this paper, we introduced the tool OCπ. This tool enables users to load object-centric event data and explore novel insights: The process execution variants contained in the object-centric event data and their frequencies. Furthermore, we enable the user to filter out infrequent variants, infrequent activities, and unwanted object types to discover an object-centric Petri net according to the chosen settings. This allows a user to interactively explore an object-centric process model and its most frequent variants.

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

[3] GraphViz needs to be installed. See: `https://graphviz.org/download/`
[4] `https://github.com/gyunamister/ocpa`

References

1. van der Aalst, W.M.P.: Process mining: Data science in action. Springer (2016). https://doi.org/10.1007/978-3-662-49851-4
2. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11724, pp. 3–25. Springer (2019). https://doi.org/10.1007/978-3-030-30446-1_1
3. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020). https://doi.org/10.3233/FI-2020-1946
4. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, Netherlands, October 31 - Nov. 4, 2021. pp. 128–135. IEEE (2021). https://doi.org/10.1109/ICPM53251.2021.9576886
5. Berti, A., van der Aalst, W.M.P.: Extracting multiple viewpoint models from relational databases. In: Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13-14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 379, pp. 24–51. Springer (2019). https://doi.org/10.1007/978-3-030-46633-6_2
6. Berti, A., van Zelst, S.J., van der Aalst, W.M.P.: Process mining for python (pm4py): Bridging the gap between process- and data science. CoRR **abs/1905.06169** (2019), `http://arxiv.org/abs/1905.06169`
7. Calvanese, D., Montali, M., Estañol, M., Teniente, E.: Verifiable UML artifact-centric business process models. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (eds.) Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014. pp. 1289–1298. ACM (2014). https://doi.org/10.1145/2661829.2662050
8. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Eng. Bull. **32**(3), 3–9 (2009)
9. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, Second Edition. Springer (2018). https://doi.org/10.1007/978-3-662-56509-4
10. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**(1-2), 109–141 (2021). https://doi.org/10.1007/s13740-021-00122-1
11. Fahland, D.: Describing behavior of processes with many-to-many interactions. In: Donatelli, S., Haar, S. (eds.) Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11522, pp. 3–24. Springer (2019). https://doi.org/10.1007/978-3-030-21571-2_1
12. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A standard for object-centric event logs. In: 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. Communications in Computer and Information Science, vol. 1450, pp. 169–175. Springer (2021). https://doi.org/10.1007/978-3-030-85082-1_16
13. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. Int. J. Softw. Tools Technol. Transf. **9**(3-4), 213–254 (2007). https://doi.org/10.1007/s10009-007-0038-x

14. Lomazova, I.A., Mitsyuk, A.A., Rivkin, A.: Soundness in object-centric workflow Petri nets. CoRR **abs/2112.14994** (2021), `https://arxiv.org/abs/2112.14994`
15. Park, G., van der Aalst, W.M.P.: Realizing a digital twin of an organization using action-oriented process mining. In: 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, Netherlands, October 31 - Nov. 4, 2021. pp. 104–111. IEEE (2021). https://doi.org/10.1109/ICPM53251.2021.9576846
16. Park, G., Adams, J.N., van der Aalst, W.M.P.: OPerA: Object-centric performance analysis. CoRR **abs/2204.10662** (2022). https://doi.org/10.48550/arXiv.2204.10662