Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 248, Issue 45, December 2025/January 2026, Pages 20–42
https://doi.org/10.7250/csimq.2025-45.02

Discovering Object-Centric Causal Nets with Edge Abstraction

Ednira de Moura Figueiredo* and Amin Jalali

Department of Computer and Systems Sciences, Stockholm University,
Kista, 164 25, Sweden

edfi6431@student.su.se, aj@dsv.su.se

Abstract. Object-centric process mining (OCPM) is an emerging research area that aims to analyze processes involving multiple object types (for instance, orders, items, and deliveries in an order-handling process) with complex intertwined relations captured in a richer format than traditional event logs. The richness of these data, as represented in the Object-Centric Event Log (OCEL) standard, often causes existing discovery algorithms to generate models overloaded with information, exceeding the cognitive limits of users, and reducing their practical usefulness. To address this challenge, we introduce Object-Centric Causal Nets (OCCN) together with an edge-abstraction technique that simplifies the discovered model by merging similar flows across object types. While OCCN provides native support for concurrency and choice, the edge abstraction is essential for reducing visual clutter and producing simpler yet expressive models. A Python implementation is provided, and a comparative evaluation against Object-Centric Petri Nets and Object-Centric Directly-Follows Graphs shows that OCCN with edge abstraction yields models that are easier to understand and more effective in enabling users to identify workflow patterns.

Keywords: Object-Centric Process Mining, Process Discovery, Causal Nets.

1 Introduction

Process mining enables data-driven analysis of business processes by deriving process models from event logs. Such models capture how processes actually unfold in practice and help organizations diagnose inefficiencies, verify compliance, and drive operational improvements based on evidence rather than intuition [1].

Traditional process mining techniques, however, rely on a single-case perspective: they reconstruct behavior by projecting events onto one object type (e.g., an order). While effective for simple processes, this approach breaks down when multiple interacting object types coexist. Consider an order-handling scenario in which an order contains several items. From the item perspective, the activity “create order” appears multiple times, even though it occurred only once. This distortion results from flattening the event data and disregarding the interplay between objects [2].

* Corresponding author

Reference: E. M. Figueiredo and A. Jalali, “Discovering Object-Centric Causal Nets with Edge Abstraction”, Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 45, pp. 20–42, 2025. Available: https://doi.org/10.7250/csimq.2025-45.02

Additional information. Author ORCID iD: E. M. Figueiredo – https://orcid.org/0009-0009-1161-9283, A. Jalali – https://orcid.org/0000-0002-6633-8587, PII S225599222500248X. Received: 16 October 2025. Accepted: 8 December 2025. Available online: 31 December 2025.

Object-centric process mining (OCPM) addresses this fundamental limitation. Instead of projecting events onto a single case notion, OCPM treats events as potentially related to multiple objects and acknowledges that objects themselves may be interconnected. With the introduction of the Object-Centric Event Log (OCEL) standard [3], events, objects, and their relationships can now be recorded in a structured and unified manner, enabling richer and more accurate analyses of complex processes.

Despite these advances, discovering models from object-centric data remains challenging. Existing approaches, such as Object-Centric Petri Nets (OCPN) [4] and Object-Centric Directly-Follows Graphs (OC-DFG) [5], often generate models that are visually dense and cognitively overwhelming. Their structural complexity hinders interpretability for business stakeholders [6], undermining one of the core objectives of process mining: providing clear and actionable insights.

To address this issue, this article contributes to the current body of knowledge by (i) proposing a novel method for discovering Object-Centric Causal Nets (OCCN), enabling process discovery with native support for concurrency, choice, and object-centric semantics, (ii) applying an edge-coarse-graining technique for process model simplification, reducing redundancy and improving the interpretability of discovered models, (iii) providing tool support through a Python-based implementation to enable discovery of OCCN models, allowing reproducibility and adoption, and (iv) conducting a comparative user study based on the Technology Acceptance Model (TAM) [7], showing that OCCN models are perceived as significantly more intuitive and useful than existing Object-Centric Petri Nets.

In this extended version of our earlier work [8], presented at the International Conference on Perspectives in Business Informatics Research (BIR), we go beyond the initial user acceptance analysis. We provide a comprehensive evaluation of model complexity, discovery performance, and structural characteristics, comparing OCCN against state-of-the-art object-centric discovery techniques using publicly available event logs. The results show that OCCN yields models that are easier to understand, more compact, and more effective for identifying workflow patterns.

The remainder of this article is structured as follows. Section 2 reviews foundational concepts and related work on Causal Nets. Section 3 outlines our methodology. Section 4 details the OCCN discovery and visualization approach. Section 5 presents the empirical evaluation results, and Section 6 discusses limitations and directions for future research. Finally, Section 7 concludes the article.

2 Background

This section provides the necessary background on Causal Nets (C-nets) [9], which form the conceptual basis for our approach to discovering object-centric Causal Nets.

C-nets were specifically designed for process discovery to overcome limitations inherent in traditional process modeling languages, such as internal inconsistencies and rigid execution semantics. Unlike languages that prescribe fixed execution paths, C-nets focus on capturing the causality between activities, resulting in models that are inherently more flexible and simpler [9]. These characteristics make C-nets particularly suitable for settings where processes are to be discovered directly from event data.

Figure 1 illustrates a C-net for a trip booking process [9]. Activities are connected through arcs, each annotated with black dots that indicate input and output bindings. A binding represents a combination of activities that must occur directly before (input) or may occur directly after (output) a given activity. These bindings determine the possible execution contexts for each activity and encode both alternative and parallel pathways.

For instance, in the example, activity *a* has four possible output bindings: *b*, *c*, *bd*, and *bcd*. Notably, *cd* is not a valid output binding for *a*, indicating that booking a car and a hotel without booking a flight is not allowed. Bindings with more than one dot represent an AND-split (parallel

execution), such as *bd*, while bindings with a single dot represent one possible outcome, such as *b* or *c*. The existence of multiple bindings expresses an XOR-split (exclusive choice) among these bindings – e.g., activity *a* can be followed by one of the available outgoing bindings.

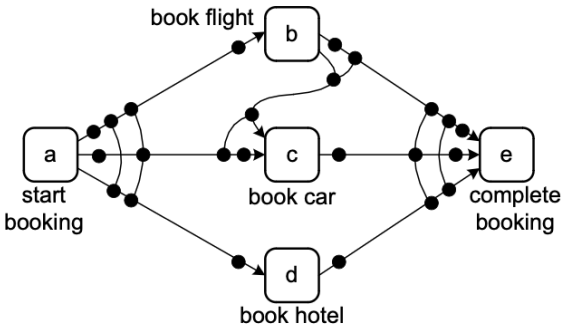


Figure 1. A C-nets model showing a trip booking process (taken from [9])

C-nets can be enriched with additional annotations that support analysis and model refinement. Activity frequencies indicate how often activities occur, binding frequencies capture how frequently specific binding materialize, and dependency measures quantify the strength of causal relations [9], [10]. These metrics help filter out accidental or noisy correlations and enhance the fidelity of the resulting models.

Due to their balance of expressive power and conceptual simplicity, C-nets are widely applied both as intermediate representations within discovery algorithms and as final modeling artifacts. Several influential process mining techniques (including heuristic miners, the Genetic Miner, and the Fuzzy Miner) build upon the principles of C-nets [1], [9]. Moreover, C-nets can be systematically transformed into Petri nets or other modeling notations when needed, supporting integration with downstream analysis tools and execution environments. Recent work further highlights their usefulness in supporting digital transformation initiatives in public-sector contexts [11].

Taken together, these characteristics make C-nets a natural foundation for extending process discovery to object-centric settings. Their ability to natively represent concurrency, choice, and causal structure aligns well with the requirements of object-centric process mining, where models must capture complex interactions across heterogeneous object types without sacrificing interpretability.

3 Research Method

In this study, we adopted the Design Science Research (DSR) framework [12], which structures the research process into five activities: problem explication, requirements definition, artifact design and development, artifact demonstration, and artifact evaluation.

During the *Problem Explication* and *Requirements Definition* phases, we conducted a comprehensive document analysis to identify gaps in existing object-centric discovery techniques and derive the requirements for a new approach. The literature review examined established discovery algorithms and assessed their suitability for object-centric contexts [4], [10], [13], [14], [15], [16], [17], [18]. In parallel, we performed a tool analysis to examine existing software artifacts in the domain. This analysis provided insights into the capabilities, limitations, and design choices of prior tools, informing the specification of functional and non-functional requirements for our artifact. Further details can be found in [19].

In the *Artifact Design* and *Development* phase, we iteratively designed and implemented the discovery and visualization algorithms. The iterative process facilitated continuous refinement of conceptual and technical choices, resulting in both prescriptive knowledge (embodied in

the implemented artifact) and descriptive knowledge (documented rationales underlying design decisions) [12].The final artifact is an instantiation in the form of a Python-based tool capable of discovering and visualizing OCCNs from OCELs.

During *Artifact Demonstration*, we applied the developed tool to discover OCCNs and compared the resulting models with OCPNs generated from the same event data. This comparison served to illustrate the artifact's functionality and highlight differences in model representation.

In the *Artifact Evaluation* phase, we assessed the perceived usefulness and perceived ease of use of the proposed approach using the TAM [7]. Participants were asked to analyze two models (an OCPN and an OCCN) and identify various workflow patterns. Their performance, along with qualitative feedback, provided insights into the interpretability, usability, and practical value of the artifact. In addition, we compared the performance of the developed artifact with current implementations, discovering OCPNs and OC-DFGs, assessing differences in model complexity and runtime efficiency.

4 OCCN Discovery and Visualization Approach

4.1 Discovering Object-Centric Causal Nets

We define a three-step method for discovering OCCN models. *First*, we discover individual C-ets for each object type by applying process discovery techniques to flattened event logs. *Second*, we merge these nets and refine their visual representation to highlight bindings and distinguish object types. *Third*, we apply an edge-aggregation procedure to remove redundant or low-value edges, thereby improving model clarity and interpretability.

4.1.1 Step 1. Discovering Individual Causal Nets

To discover OCCNs, we first flatten the event log for each object type and apply the discovery algorithm independently, resulting in one C-net per object type. Flattening refers to projecting an object-centric event log onto a single object type and transforming it into a traditional single-case event log. This approach aligns with prior work [4], [5], which emphasizes methodological simplicity and leverages existing discovery techniques on flattened logs.

Consider an order management scenario with object types such as 'orders', 'items', 'products', 'packages', and 'customers' in an order management process. A 'customer' object may 'place' multiple 'order' objects. By flattening the log per object type, we obtain separate traditional event logs for each type and can apply heuristic discovery techniques to derive corresponding C-nets.

For each object type, a C-net is mined heuristically by identifying bindings that represent AND, XOR, and OR splits and joins. This follows replay-based procedures described in [17] and the Flexible Heuristics Miner [15]. C-nets distinguish between two categories of bindings: single bindings and compound bindings, as illustrated in Figure 2. Single bindings capture XOR splits and joins, whereas compound bindings capture AND-split and AND-join behavior.

For instance, the input binding preceding "create package" with a frequency of 21 is a single binding, indicating that this activity followed "pay order" without "pick item" occurring, representing an XOR-join. In contrast, another input binding with a frequency of 33 is a compound binding, showing that "create package" occurred only after both "pay order" and "pick item" had taken place, representing an AND-join.

4.1.2 Step 2. Merging C-Nets and Refining the Model:

In this step, we apply three procedures: (i) merging the C-nets discovered for each object type, (ii) distinguishing input and output bindings using distinct visual notations to improve readability, and (iii) applying color coding to differentiate object types within the integrated model.

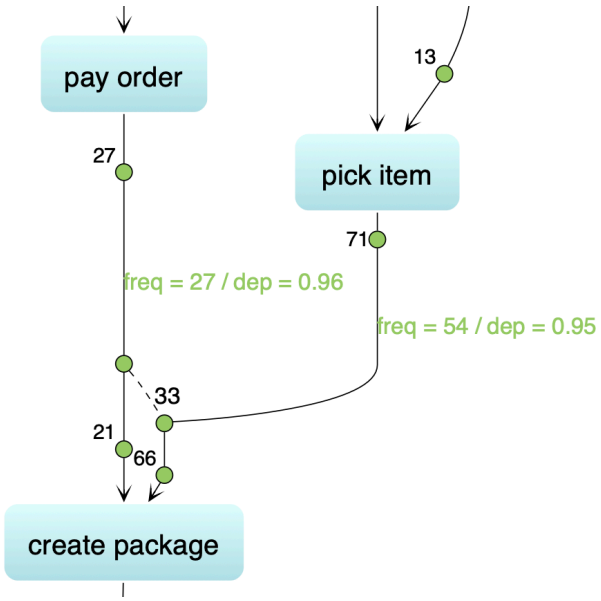


Figure 2. An excerpt from a sample C-nets illustrating both single and compound bindings identified during the initial discovery step of the algorithm.

First, our algorithm merges all individual C-nets into a unified representation. While this integration represents the process flow from several objects' perspectives, it may also introduce substantial visual complexity due to the large number of bindings involved. Such complexity can hinder the interpretability of the resulting model. Therefore, after the initial merge, we apply two additional refinement steps aimed at simplifying the visualization.

To clearly distinguish between input and output bindings, output bindings are rendered as circles, while input bindings are rendered as diamonds. As illustrated in Figure 3, the activity "reorder item" has an input binding (diamond) originating from "item out of stock," indicating that "item out of stock" precedes "reorder item." Conversely, the output binding for "item out of stock" (circle) shows that it is followed by "reorder item." This visual differentiation can help users quickly recognize causal relationships and interpret model semantics more effectively.

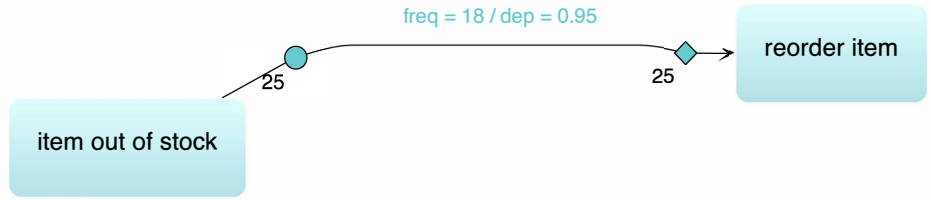


Figure 3. An excerpt from the discovered OCCN after applying modified visual encoding for input and output bindings in the second step of the algorithm.

To differentiate between object types within the merged model, our algorithm assigns distinct colors to bindings according to their associated object type. As illustrated in Figure 4, single bindings belonging to the customer types are shown in red, bindings for orders appear in yellow, and both single and compound bindings for items are depicted in purple. In this example, two AND-split patterns (with frequencies of 125 and 26) are visible for the items object type following the "place order" activity, highlighting parallel behavior specific to this object class.

4.1.3 Step 3. Aggregating Single Bindings:

Existing object-centric discovery algorithms often produce highly complex models that are difficult for business stakeholders to interpret [6], [20]. A major contributor to this complexity is the

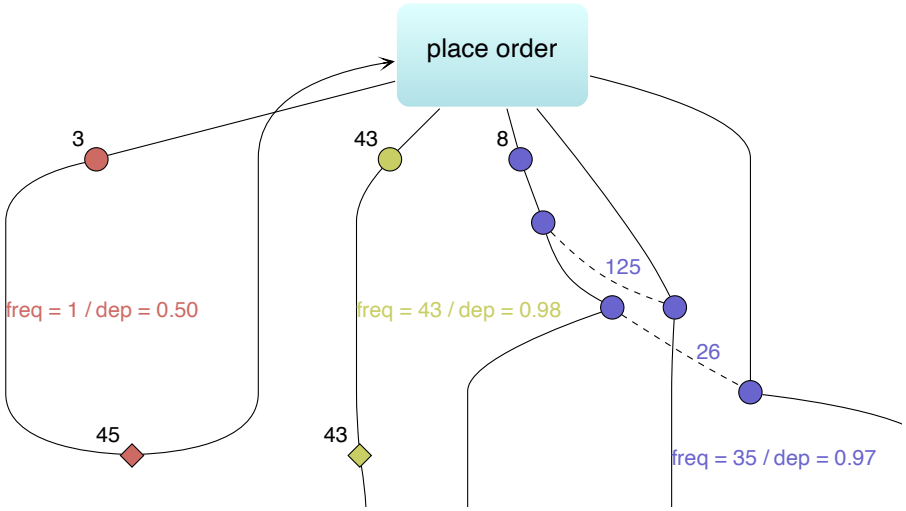


Figure 4. An excerpt from a sample OCCN illustrating the structure resulting from the second step of the discovery algorithm.

presence of redundant flows, i.e., multiple edges that express the same ordering relations between activities but originate from different object types. To address this issue, coarse-graining techniques have been proposed in the broader field of graph simplification. In particular, edge-coarse-graining offers a principled way to reduce graph complexity by consolidating structurally similar edges [21], [22]. Adjusting the level of granularity in this manner can make process models more effective in revealing meaningful behavioral patterns [23], [24], [25], [26].

In our approach, we apply edge-coarse-graining by merging edges representing single bindings that share identical source and target activities. This consolidation removes redundant edges while preserving the essential causal structure of the model. Figure 5 illustrates this morphism, denoted as λ, which maps multiple fine-grained edges onto a single coarse-grained edge. In the example, edges e_1 and e_2 are mapped to h_1, while e_3 and e_4 are mapped to h_2, resulting in $\lambda(e_1) = \lambda(e_2) = h_1$ and $\lambda(e_3) = \lambda(e_4) = h_2$. Through this aggregation process, the model's visual complexity is reduced without sacrificing the representation of underlying workflow relations, thereby enhancing interpretability.

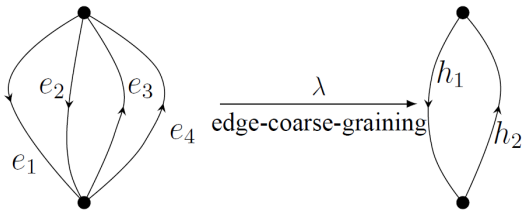
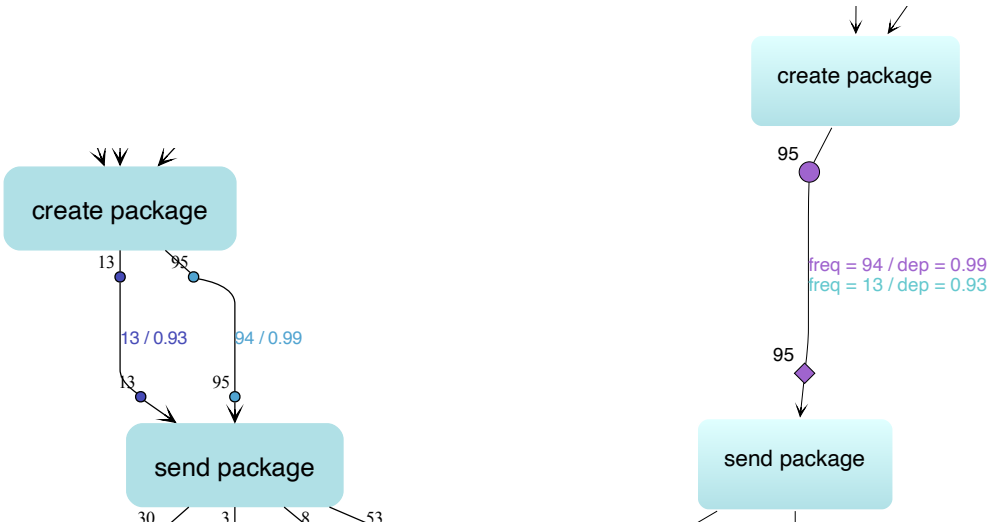


Figure 5. Edge-coarse-graining technique to merge flows (adapted from [27])

Figure 6 illustrates the edges between the activities "create package" and "send package" before (Figure 6a) and after (Figure 6b) applying edge merging. Following the merge, the algorithm creates an aggregated object type that stores the attributes of all contributing edges. In this example, the frequency and dependency measures of the two original edges are transferred to the newly created aggregated edge, allowing users to inspect the underlying contributions of each object type when needed.

Finally, the algorithm enriches the OCCN model with descriptive statistics, including the mean, median, minimum, and maximum number of occurrences of each activity across different object types. Rather than cluttering the visual model with these details, they are made available through interactive tooltips that appear when a user hovers over an activity node. This design

choice balances information richness and visual simplicity, supporting a user-centered visualization approach. Maintaining simplicity is crucial for enhancing understandability in process mining models [1], and it is particularly important in object-centric scenarios where model complexity can grow quickly [5]. Figure 7 illustrates this by showing the number of objects linked to the place order activity for each object type. Further details about the approach can be found in [19].



(a) An excerpt from the discovered OCCN before applying the edge-coarse-graining technique.

(b) An excerpt from the discovered OCCN after applying the edge-coarse-graining technique.

Figure 6. Illustration of a discovered OCCN model before and after applying the edge-coarse-graining technique.

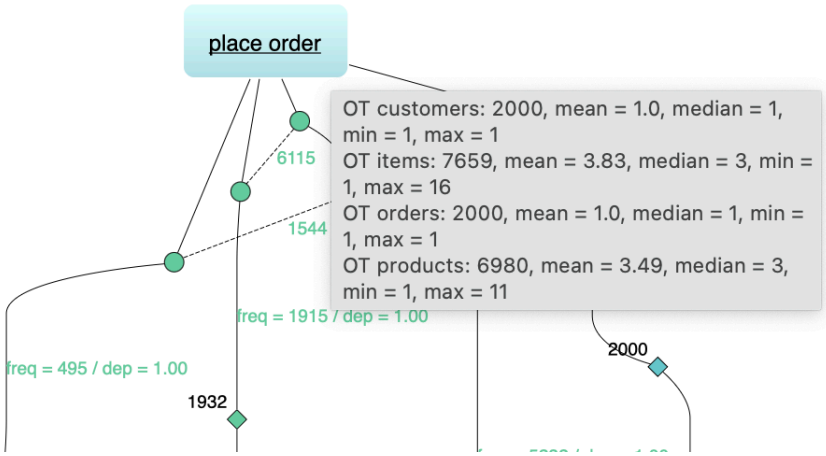


Figure 7. An excerpt from the discovered OCCN illustrating the use of tooltips to provide additional detail on demand.

4.2 Implementation

We implemented two algorithms in Python for the discovery and visualization of OCCNs. To ensure transparency and reproducibility, the implementation is publicly available on GitHub[†] together with illustrative examples. To support adoption by diverse stakeholders in both business and academia, the design deliberately avoids reliance on command-line prompts and complex installation procedures, which could otherwise discourage use by less experienced users.

[†] https://github.com/ednira/cnets_project

The user interface is provided through a Jupyter Notebook environment, enabling guided interaction with the code. Users are only required to specify the path to the event log file, while the selection of object types and dependency measures remains optional and can be customized as needed. The discovered process model is rendered as an interactive SVG within a notebook cell, supporting engagement through mouse-hover tooltips that display activity-related statistics. The intended user profile, therefore, includes familiarity with Jupyter Notebooks, which offer usability advantages over command-line interfaces for exploratory and visual analysis. Figure 8 illustrates the use of our implementation; the figure is not intended to be readable and is provided solely for illustrative purposes.

Although this study focuses on model simplicity, we also evaluate the performance of our implementation to assess scalability. Experiments were conducted on a professional-grade laptop equipped with an 8-core CPU (comprising 6 performance cores and 2 efficiency cores) and 16 GB RAM. Table 1 reports statistics comparing our technique with OCPN and OC-DFG across eight publicly available event logs, while Figure 9 provides a side-by-side comparison of the three algorithms. These logs contain between 3 and 12 object types (average = 7.3), 8 to 24 activities (average = 13.6), and 598 to 38 528 events (average = 21 402). Runtime in seconds was measured for OCCN, OCPN (PM4Py[‡]), and OC-DFG (PM4Py). The results show that the runtime of OCCN increases only marginally, from 1.946 seconds for the CargoPickup log to 2.236 seconds for the Order Management log, which contains more than 35 times as many events. Overall, the comparison across the eight logs demonstrates that OCCN scales effectively, although runtime may vary depending on the number of object types, activities, and events.

Table 1. Performance comparison of OCCN, OCPN, and OC-DFG (runtime in seconds)

Log	OT	Objects	e2o	Events	Act.	Runtime OCCN	Runtime OCPN	Runtime OC-DFG	Ratio OCPN OCCN	Ratio OC - DFG OCCN
Order Mgmt [28]	6	10 840	147 385	21 008	11	2.236	1.741	1.550	0.779	0.693
Logistics [29]	7	13 882	74 272	35 372	14	1.789	1.250	1.869	0.699	1.045
P2P [30]	7	9054	35 927	14 671	10	0.902	0.575	0.787	0.637	0.872
Hinge [31]	12	23 771	144 827	38 528	11	3.470	8.669	2.589	2.499	0.746
CargoPickup [32]	3	70	3457	598	8	1.946	0.294	0.115	0.151	0.059
Hiring [33]	6	3768	34 908	18 119	24	1.029	0.486	0.881	0.472	0.857
Hospital [33]	8	3688	27 605	14 642	9	0.494	0.424	0.727	0.858	1.472
Order-to-Cash [33]	9	8819	55 360	28 278	22	1.324	0.886	1.439	0.669	1.087

Our findings indicate that OCPN (PM4Py) model discovery requires, on average, 85% of the OCCN runtime (see column ‘Ratio OCPN OCCN’). For the Hinge process, OCCN discovery is more efficient than OCPN, taking only 40% of the time required by OCPN. In contrast, for the CargoPickup process, OCPN demonstrates greater efficiency relative to OCCN compared to other processes. On average, OC-DFG runtime corresponds to 54% of OCCN runtime (see column ‘Ratio OC-DFG OCCN’) and outperforms both OCPN and OCCN, as it does not mine concurrency or choice splits and joins. Since these metrics are expressed in seconds, the differences are relatively small. Nevertheless, further evaluation is needed to obtain a more comprehensive understanding of performance. The advantages of simplicity and expressiveness offered by OCCN may compensate for these trade-offs.

[‡] PM4Py is a leading open-source process mining library in Python, designed for use in both academia and industry: https://processintelligence.solutions/pm4py

jupyter demonstration Last Checkpoint: 4 hours ago

File Edit View Run Kernel Settings Help Trust

Code JupyterLab Python 3 (ipykernel)

Discovering Object-Centric Causal Nets with Edge-Coarse-Graining

This notebook demonstrates the discovery and visualization of an Object-centric Casual nets with Edge-Coarse-Graining.

[1]:
```
import pm4py

from discover_occnets import *
from view_occnets_jupyter import *

file_path = './ContainerLogistics.json'
ocel, ot_activities, event_to_obj, obj_to_obj = import_log(file_path)
ot_activities, subgraphs = subgraphs_dict(file_path, dependency_threshold=0.99)
OCCN_model = all_ot_visualization(ot_activities, subgraphs, profile=['Vehicle', 'Customer Order', 'Transport Document'])
```

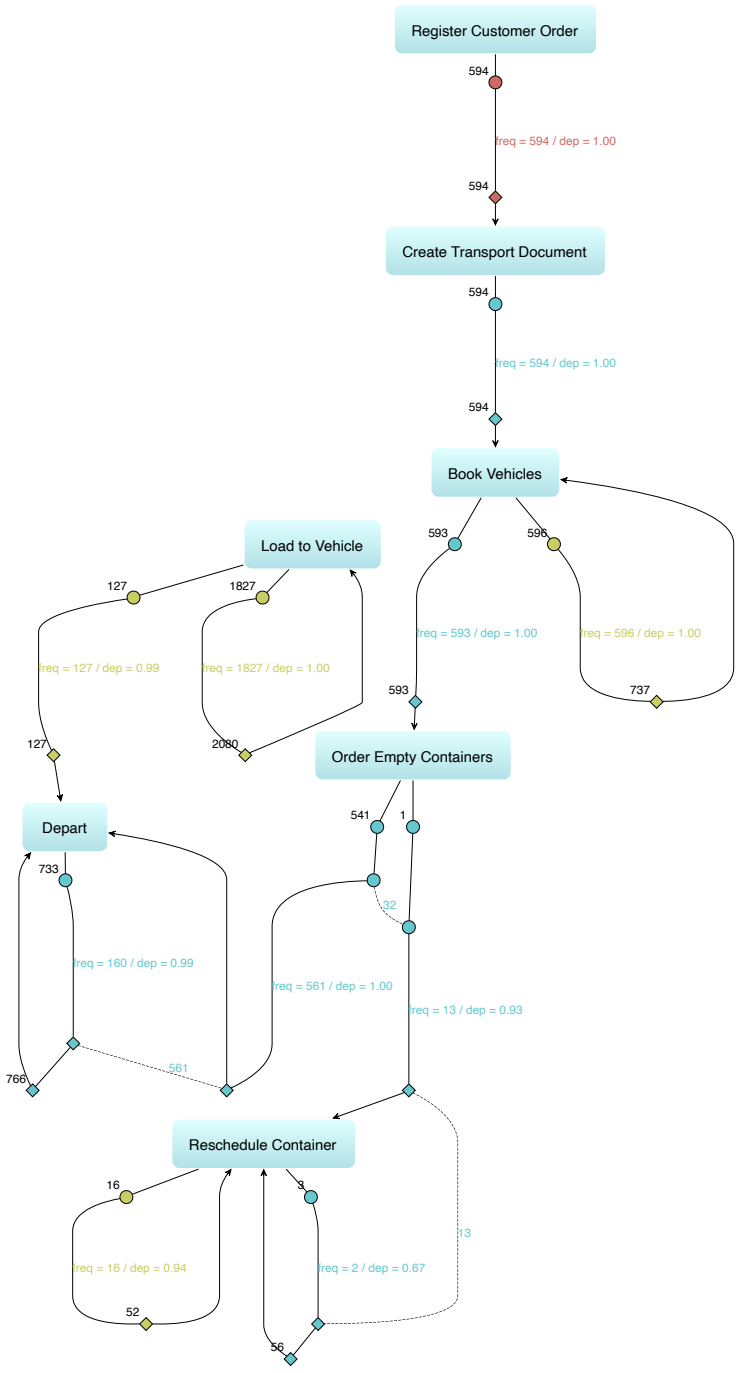


Figure 8. Demonstration of our implementation in the Jupyter Notebook environment

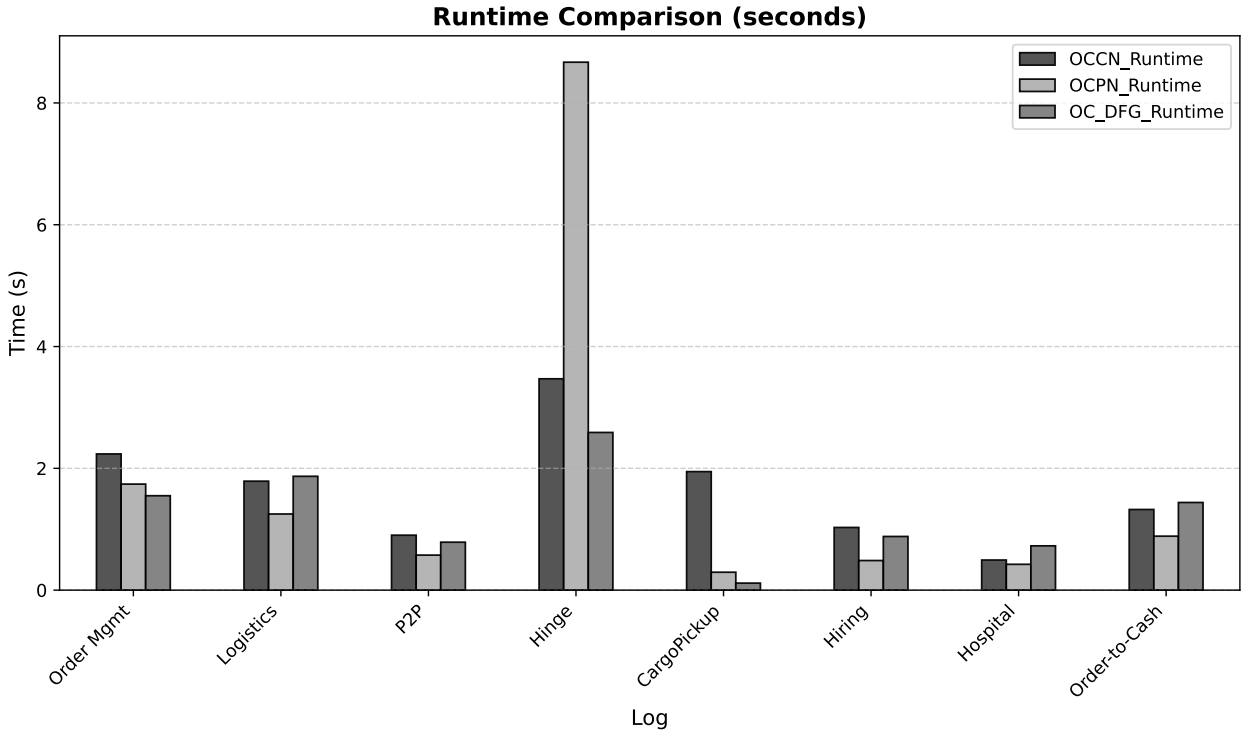


Figure 9. Side-by-side performance comparison of OCCN, OCPN, and OC-DFG, showing runtimes (in seconds) across event logs.

5 Demonstration and Evaluation

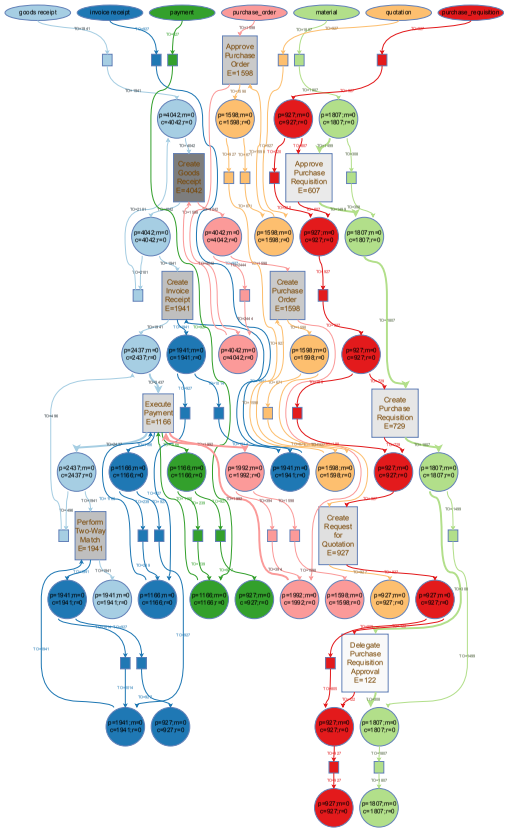
At the time of the research, two main approaches to object-centric process discovery were available: OCPN [4] and OC-DFG [5]. OCPN can capture control-flow patterns such as parallel joins/splits and exclusion joins/splits. Therefore, we use OCPN as a baseline for comparison with our approach. This section first demonstrates how OCPN models and OCCN models (discovered using our method) appear side by side, and then presents the results of a user acceptance evaluation.

5.1 Demonstration

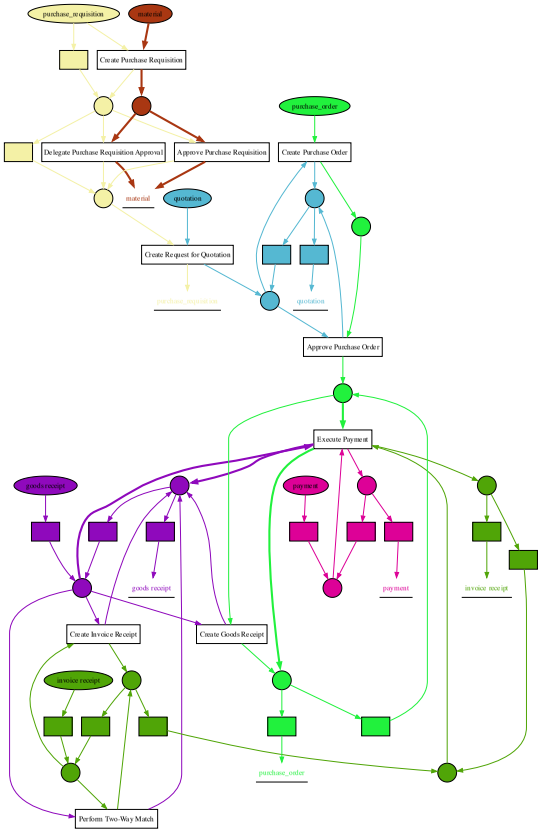
Figure 10 presents the OCPN model discovered using the OC-PM tool[§] (Figure 10a), the OCPN model discovered using PM4Py (Figure 10b), and the OCCN model discovered using our approach (Figure 10c), all based on the same publicly available log file, Procurement-to-Pay OCEL 2.0 [30]. We intentionally provide a zoomed-out version of the models here, as it allows observing the general structure and flow of the process. As can be seen, the flow of work is recognizable in the model discovered using our approach, while it is very difficult in the one discovered using the OC-PM tool. The model discovered using PM4Py is also difficult to follow due to many silent activities (the ones filled with color and without any activity names). For those interested in exploring the models in more detail, the full process models can be accessed in the GitHub repository and in [19].

The complexity of process models can be measured using different techniques that mainly consider the number of different control-flow elements within the model. As none of the metrics are adapted for object-centric process models, we counted the number of different elements and compared the total numbers, indicating estimates on the complexity of each discovered model. Table 2 presents the number of different types of visual elements, such as object type, activity, arc,

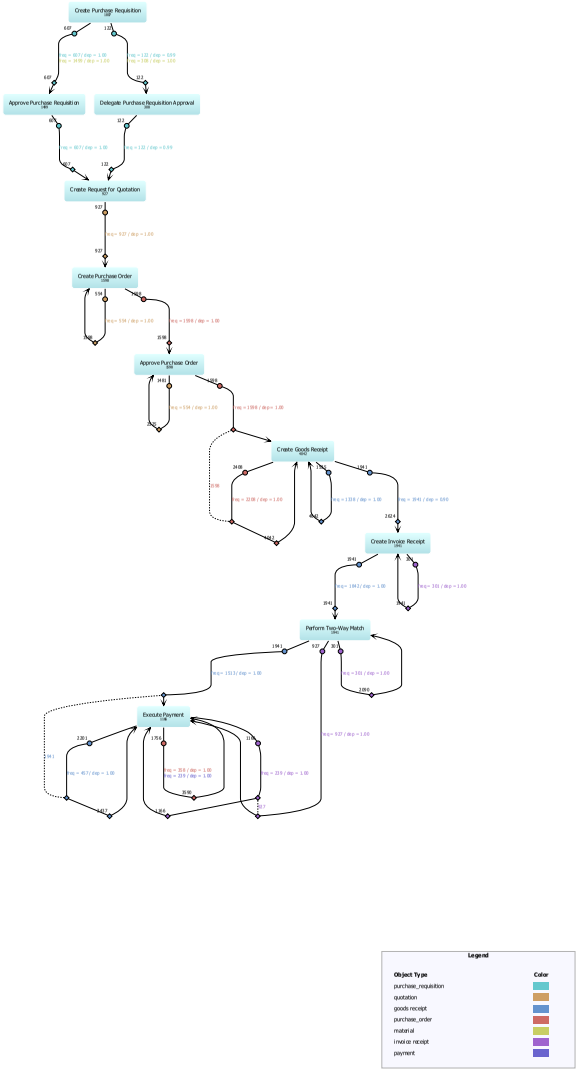
[§] Berti, A., van der Aalst, W.M.P., "OC-PM: analyzing object-centric event logs and process models," *International Journal on Software Tools for Technology Transfer*, vol. 25, pp. 1–17, 2023. Available: https://link.springer.com/article/10.1007/s10009-022-00668-w



(a) OCPN mined with OC-PM



(b) OCPN mined with PM4Py



(c) OCCN mined with our approach

Figure 10. Demonstrating OCPN and OCCN models discovered from Procurement-to-Pay OCEL 2.0

place, binding, and silent transition, for each model. As can be seen, OCCN models incorporate a smaller number of total visual elements, indicating these models might be less complex for users to comprehend. We have evaluated how users perceived these models as described in Section 5.2.

Table 2. Number of visual elements exists in discovered object-centric process models

	Object Types	Activities	Arcs	Places	Bindings	Silent Transitions	Total
OCPN (OC-PM)	7	10	104	37	0	30	188
OCPN (PM4Py)	7	10	78	16	0	17	128
OCCN (our approach)	7	10	20	0	40	0	77

5.2 Evaluation Design

We evaluated our method using both quantitative and qualitative approaches. The evaluation focused on two main aspects: (i) user comprehension, assessed through pattern recognition tasks, and (ii) user acceptance, measured using the TAM [7]. In addition, open-ended feedback was collected to inform potential improvements. A comparative survey-based approach was employed. We used a structured questionnaire to collect responses from 17 students in the Department of Computer and Systems Sciences at Stockholm University (DSV/SU).

Among the participants, 70.59% were bachelor's students and 29.41% were master's students. Regarding professional experience, 41.18% had at least two years of industry experience, and 23.53% reported prior knowledge of process modeling languages. Participants were also asked about their online shopping frequency, as the models examined in the study referred to an order management process. A large majority (94.12%) indicated that they shop online, with 70.59% doing so approximately once per month.

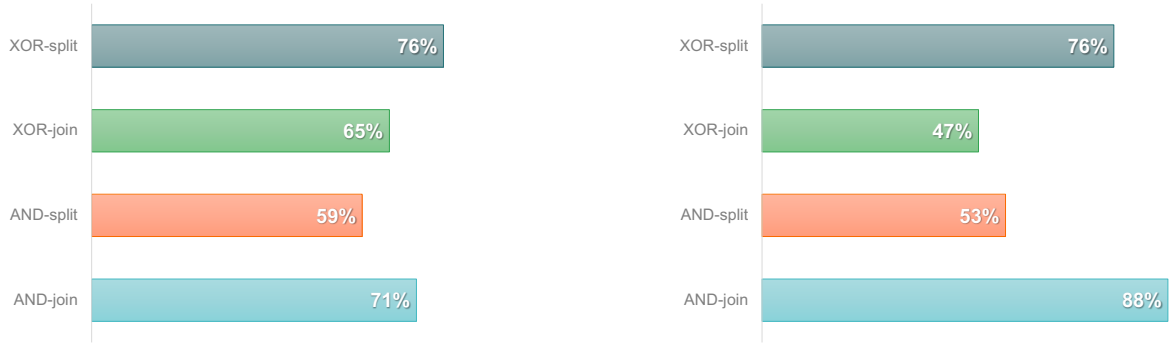
Participants analyzed both an OCCN model (discovered using our method) and an OCPN model (discovered using the OC-PM tool). Both models were based on the same log file, namely the Order Management Object-Centric Event Log [28]. This log file was selected because many participants, regardless of background, were likely to have experience with online ordering, making the process familiar to them. To minimize bias, participants were divided into two groups: one starting with the OCCN model and the other with the OCPN model. The questionnaire consisted of three sections:

- Pattern recognition task: Participants identified control-flow patterns in the models through both open-ended and multiple-choice questions.
- Perceived usefulness (PU) and perceived ease of use (PEU): These were measured for each model using TAM.
- Open-ended feedback: Participants reflected on the strengths and challenges of both models.

5.3 Pattern Recognition

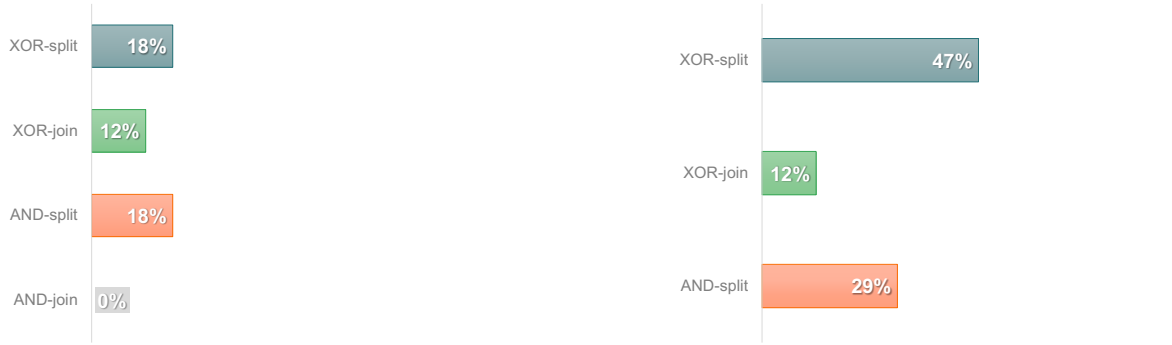
In the first part of the questionnaire, participants answered two sets of questions. The first set consisted of open-ended questions designed for the pattern recognition task, which helped us assess their understanding of the model after the first author explained the patterns using diagrams included in the questionnaire. Respondents reported that this was the most challenging part of the questionnaire. The second set comprised multiple-choice questions, where participants were shown parts of the model and asked to identify specific patterns.

As shown in Figure 11, most respondents correctly identified all four patterns in the OCCN model, with balanced results across both question types. Specifically, 76% of respondents correctly recognized the XOR-split in the open-ended questions, while 88% correctly identified the AND-join in the multiple-choice questions. In contrast, Figure 12 presents the results for the OCPN model, where participants struggled to recognize the patterns satisfactorily.



(a) Using Open-ended Questions **(b)** Using Multiple-choice Questions

Figure 11. The ratio of correctly identified workflow patterns using the discovered OCCN model by participants



(a) Using Open-ended Questions **(b)** Using Multiple-choice Questions

Figure 12. The ratio of correctly identified workflow patterns using the discovered OCPN model by participants

5.4 User Acceptance

We used the TAM [7] in the second part of the questionnaire to evaluate the user acceptance, which is one widely used method in evaluating information systems artifacts in general and business process-related artifacts in particular [34], [35], [36], [37]. To measure PU and PEU, we used questionnaires to collect participants' perceptions through scales of: extremely unlikely (1), quite unlikely (2), slightly unlikely (3), neither likely nor unlikely (4), slightly likely (5), quite likely (6), and extremely likely (7).

Figure 13 displays the aggregated distribution of responses regarding participants' perceptions of the two modeling languages – OCCN (denoted as M1) and OCPN (denoted as M2). The results reveal that both PU and PEU medians for OCCN are at least two points higher than those for OCPN. Specifically, the median PU and PEU scores for OCCN are 5.17 and 4.5, respectively, while both scores for OCPN are 2.5. In Figure 14, a more detailed comparison of these perceptions is presented. The results indicate a clear preference for OCCN, with higher scores across both dimensions (PU and PEU). However, further analysis was conducted to assess whether these differences are statistically significant.

Normality testing was performed on the responses for PU in the OCCN group, which revealed that the data followed a normal distribution except for the PU of OCCN. As a result, the non-parametric Mann–Whitney U test was employed to determine whether the PU of OCCN and OCPN differed significantly. The test yielded a p-value of 0.00178, indicating a statistically significant difference in perceived usefulness between the two models.

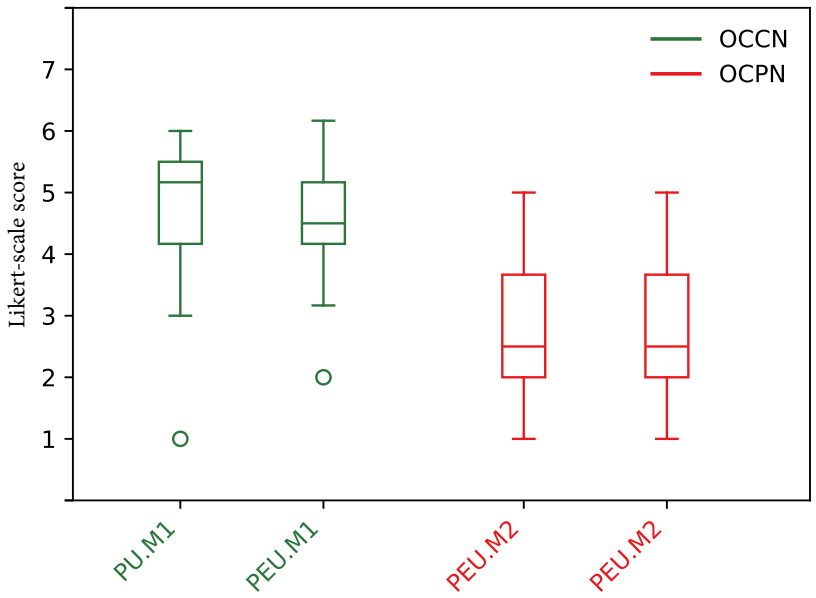
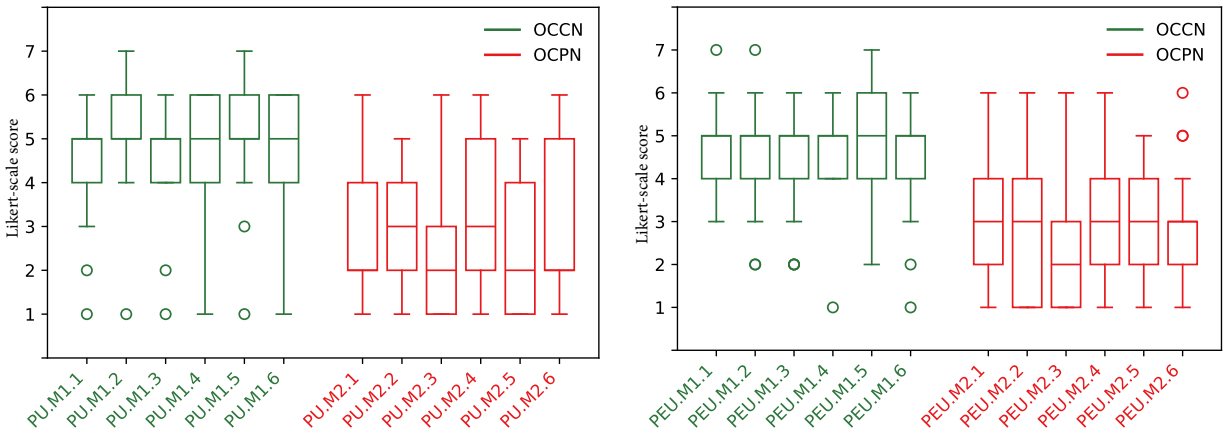


Figure 13. Aggregated PU and Aggregated PEU for OCPN and OCCN (contains colors – perception will be limited when printed/converted to grayscale).



(a) Perceived Usefulness details

(b) Perceived Ease of Use details

Figure 14. Detailed comparison of PU and PEU between OCPN and OCCN models (contains colors – perception will be limited when printed/converted to grayscale)

To evaluate the PEU, the independent samples t-test and the paired samples t-test were applied. Both tests resulted in a p-value of approximately 0.0001, confirming that the observed differences in PEU between the two models were statistically significant.

Cronbach's alpha was calculated to assess the internal consistency of the PU and PEU constructs. As shown in Table 3, the values for both PU and PEU exceeded 0.9, which is considered excellent and well above the generally accepted threshold of 0.7.

Table 3. Cronbach's Alpha for PU and PEU of both models

Language	**PU**	**PEU**
OCCN	0.95	0.90
OCPN	0.94	0.92

5.5 User Feedback Evaluation Result

The last question asked for feedback about the used models. The feedback was analyzed by splitting it into sentences and finally reducing them to words that best summarize the feedback, keeping users' words as much as possible. The list of words was loaded into an online word cloud generator [38]. The word cloud visually demonstrates the users' sentiments about object-centric modeling languages. The size of each word indicates how frequently users mentioned it. They declared a poor understanding of the OCPN model and deemed it cluttered and unintuitive. Conversely, the OCCN model was considered easy, understandable, clear, and intuitive. Figure 15 (a and b) shows users' sentiments visually.

(a) Object-centric Petri net sentiment

(b) Object-centric Causal nets sentiment

Figure 15. Word clouds showing participant sentiment for each model: (a) OCPN, (b) OCCN

5.6 Model Comparison – Entities

In this subsection, we compare the implemented OCCN discovery with the OCPN and OC-DFG algorithms from the PM4Py library. Although OC-DFG does not support choice or concurrency, we include it for a more comprehensive evaluation. Since the primary goal of this research is model simplicity, our analysis focuses on the number of edges and the number of activities discovered by each technique.

5.6.1 Experiment Setup

For this experiment, we use eight OCEL 2.0 event logs that are publicly accessible [3]. The results are shown in Table 4, with the following statistics:

- OT: number of object types in the log;
- Objects: number of objects in the log;
- Events: number of events in the log;
- Act.: number of activities in the log;
- D.A. OCCN: number of activities discovered by OCCN (our tool);
- D.A. OCPN: number of activities discovered by OCPN (PM4Py);
- D.A. OC-DFG: number of activities discovered by OC-DFG (PM4Py);
- Edges OCCN: number of edges discovered by OCCN (our tool);
- Edges OCPN: number of edges discovered by OCPN (PM4Py);
- Edges OC-DFG: number of edges discovered by OC-DFG (PM4Py);
- Edge Ratio OCPN: ratio Edges OCPN/Edges OCCN;
- Edge Ratio OC-DFG: ratio Edges OC-DFG/Edges OCCN;
- Act. Ratio OCPN: ratio D.A. OCPN/D.A. OCCN;
- Act. Ratio OC-DFG: ratio D.A. OC-DFG/D.A. OCCN.

Table 4. Discovered activities, edges, and ratios relative to OCCN for selected object-centric process discovery techniques

Log	OT	Objects	Events	Act.	D.A.			Edges			Edge Ratio		Act. Ratio	
					OCCN	OCPN	OC-DFG	OCCN	OCPN	OC-DFG	OCPN	OC-DFG	OCPN	OC-DFG
Order Mgmt [28]	6	10 840	21 008	11	11	75	11	55	152	244	2.76	4.44	6.82	1.00
Logistics [29]	7	13 882	35 372	14	14	63	14	44	128	76	2.91	1.73	4.50	1.00
P2P [30]	7	9054	14 671	10	10	39	10	26	78	52	3.00	2.00	3.90	1.00
Hinge [31]	12	23 771	38 528	11	11	88	11	37	176	107	4.76	2.89	8.00	1.00
CargoPickup [32]	3	70	598	8	8	12	8	22	24	30	1.09	1.36	1.50	1.00
Hiring [33]	6	3768	18 119	24	24	66	24	51	132	63	2.59	1.24	2.75	1.00
Hospital [33]	8	3688	14 642	9	9	45	9	27	92	38	3.41	1.41	5.00	1.00
Order-to-Cash [33]	9	8819	28 278	22	22	47	22	67	94	69	1.40	1.03	2.14	1.00

Next, we analyze the number of entities generated by the models, specifically, the number of edges and the number of activities. The extant literature shows that the number of entities in a graph can potentially impact model readability and clarity [1]. The Occam's Razor principle states that "one should not increase, beyond what is necessary, the number of entities required to explain anything" and one should aim at the simplest model able to explain the phenomena contained in a dataset [1].

5.6.2 Model Complexity – Activities

We begin by comparing the number of activities discovered by OCCN, OCPN, and OC-DFG, as this metric directly influences model clarity. Table 4 reports the number of activities present in the logs (Act.), alongside the activities discovered by each technique (D.A. OCCN, D.A. OCPN, and D.A. OC-DFG). Since OCCN and OC-DFG mine exactly the activities contained in the log, no further comparison between them is required.

In contrast, OCPN models discover substantially more activities. On average, the number of activities in OCPN (Act. Ratio = 4.32) is about four times higher than in OCCN. This difference stems from Petri net semantics: whereas OCCN models only include the activities found in the log, OCPN introduces additional non-labeled activities, known as silent transitions. Figure 16 illustrates these differences across the three techniques. For instance, in the Hinge process, the number of activities in the OCPN model is eight times greater than in the OCCN model.

OCPN uses silent transitions to represent unobservable behavior, that is, activities not recorded in the log [1]. These transitions are necessary constructs to route the flow between activities and, for instance, to enable skipping activities, thereby increasing model expressiveness [1]. The drawback, however, is increased model complexity and reduced readability. Users must trace sequences of edges and silent transitions to understand the behavior being represented, whether it involves control-flow patterns (AND, XOR) or skipped activities. In object-centric process mining, this reduction in readability becomes even more critical, as the semantics can produce cluttered models due to the relationships among multiple object types.

Figure 17 illustrates the impact of silent transitions on model readability. It shows the flow between the activities "Create Invoice Receipt," "Perform Two-Way Match," and "Execute Payment." On the left, the OCCN model (our tool) depicts a straightforward sequence: "Create Invoice Receipt" is followed by "Perform Two-Way Match," which in turn is followed by "Execute Payment." The control flow is clear and unidirectional. On the right, the same behavior is represented in the OCPN model (PM4Py). Here, the flow is not unidirectional: after "Create Invoice Receipt," it moves downward to "Perform Two-Way Match," then upward to a place positioned after "Create Invoice Receipt." The flow changes direction again, moving downward to a silent

transition, then onward to another place before finally reaching “Execute Payment.” Additionally, edges from another object type intersect with this sequence, further hindering clarity.

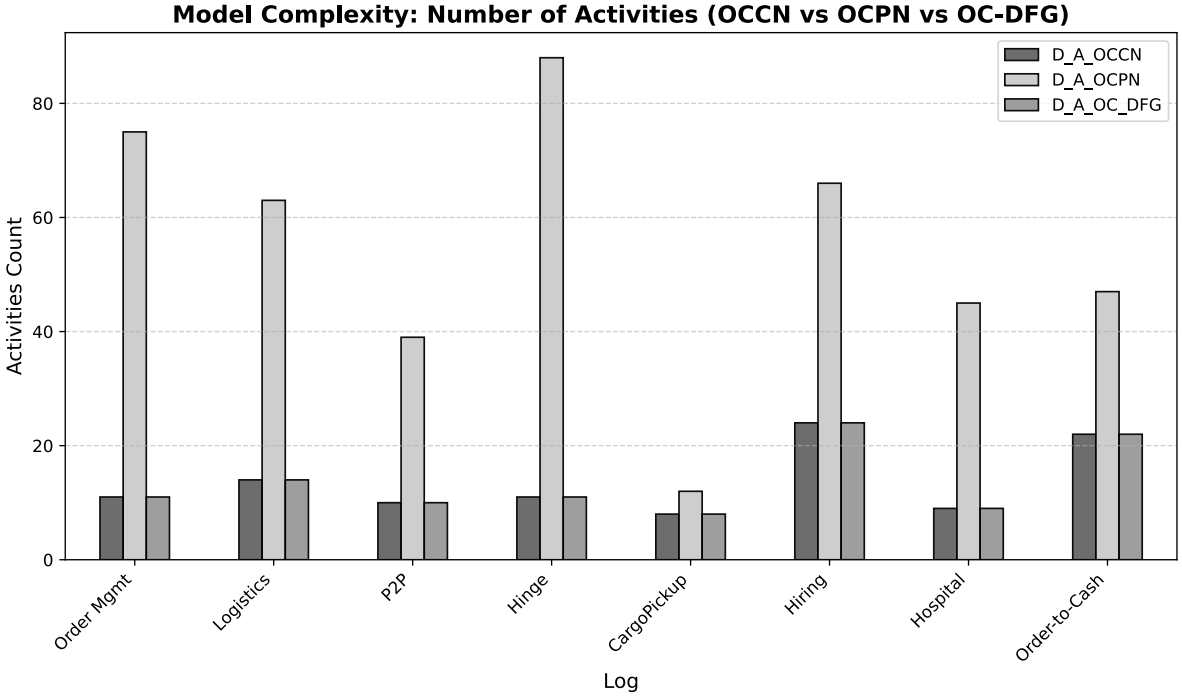
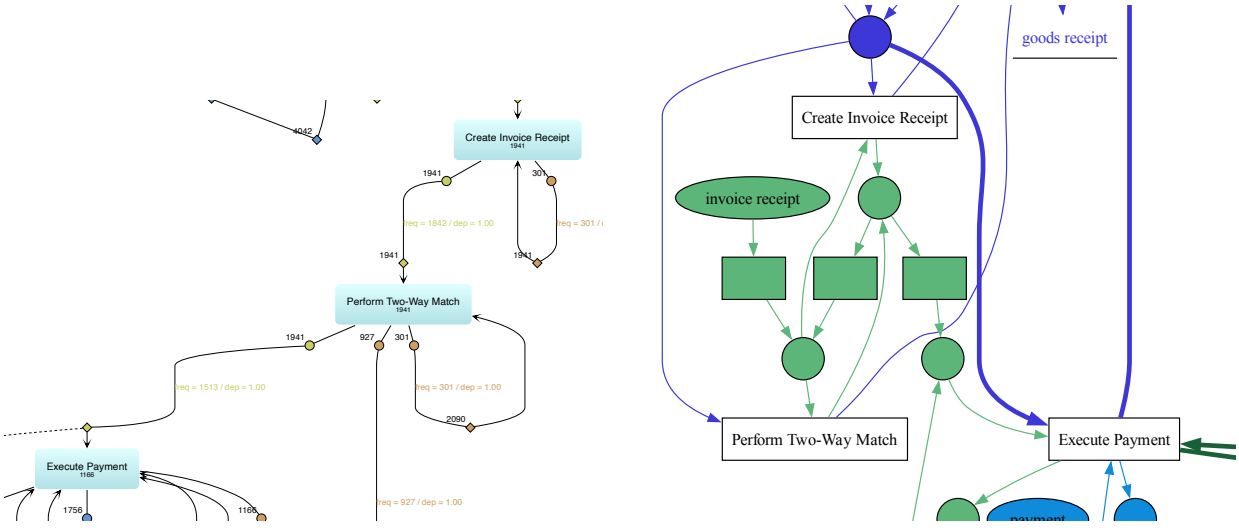


Figure 16. Comparison of discovered activities in OCCN, OCPN, and OC-DFG models



(a) OCCN (our tool) – P2P process detail - activities **(b)** OCPN (PM4Py) – P2P process detail - activities

Figure 17. Model comparison – the number of activities (silent transitions) impacts readability

5.6.3 Model Complexity – Edges

We compare the number of edges discovered by the object-centric techniques because it also influences model simplicity. The metrics ‘Edges OCCN’, ‘Edges OCPN’, and ‘Edges OC-DFG’, shown in Table 4, inform how many edges are generated by the OCCN, OCPN, and OC-DFG algorithms, respectively. Our results show that OCPN implemented in PM4Py produces, on average (avg. of ‘Edge Ratio OCPN’ = 2.74), almost three times the number of edges in comparison to

OCCN implemented with our tool. OC-DFG produces, on average (avg. of 'Edge Ratio OC-DFG' = 2.01), two times the number of edges our implementation of OCCN does. These findings show that both OCPN and OC-DFG models increase model complexity substantially in the edge dimension.

It is noteworthy that the increase in activities due to silent transitions, as discussed earlier, also leads to a corresponding increase in the number of edges in the OCPN model. Because of OCPN semantics, two elements of the same type cannot be connected directly. Instead, places must be connected to transitions and vice versa, and the presence of silent transitions further increases the number of edges in the model. In object-centric process mining, this representational bias is even more challenging than in traditional process mining, as multiple object types amplify the complexity. Figure 18 compares 'Edges OCCN,' 'Edges OCPN,' and 'Edges OC-DFG.' For instance, in the Hinge process, the number of edges in the OCPN model is nearly five times that of the OCCN model, while the OC-DFG model has almost three times as many edges as OCCN.

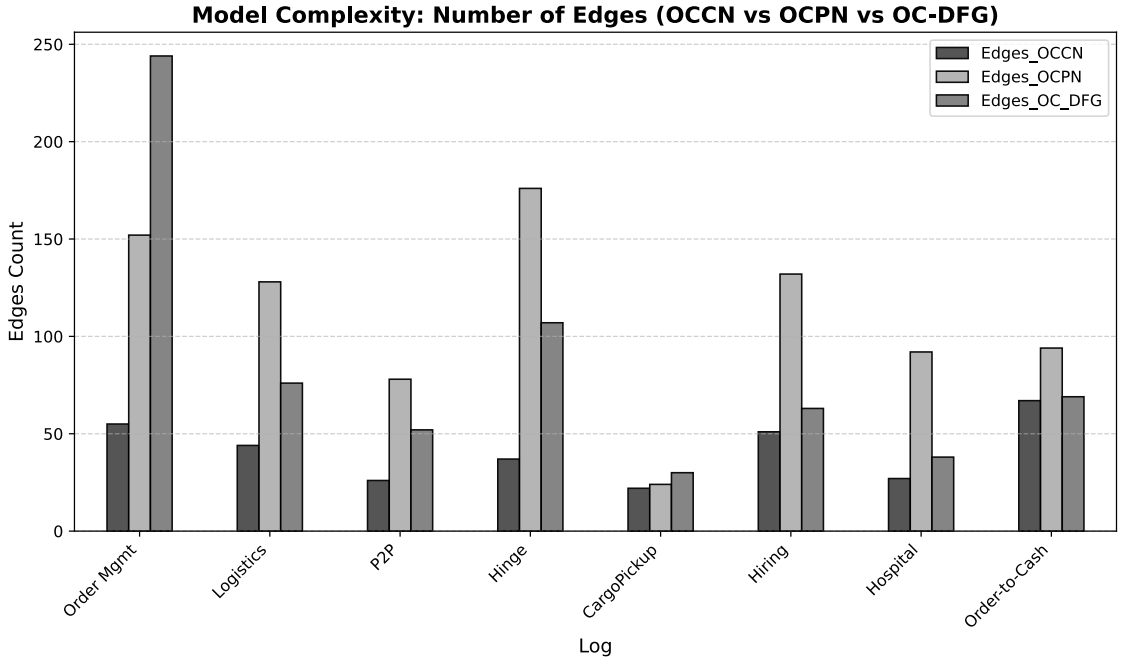
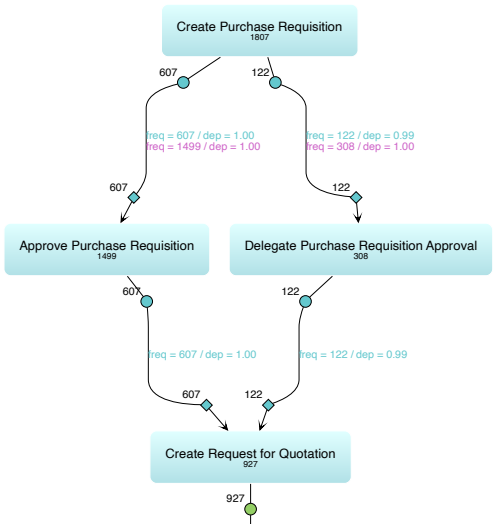


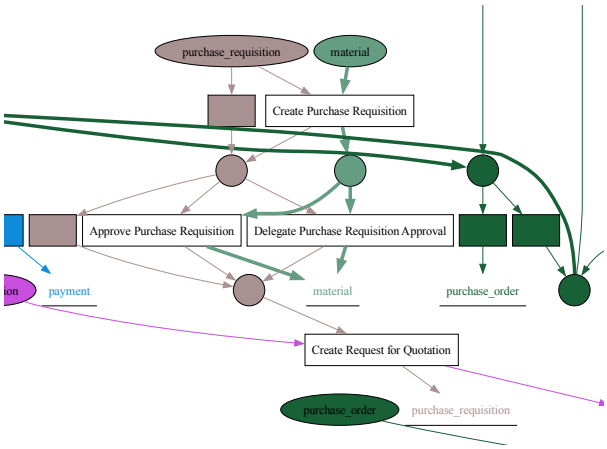
Figure 18. Edges in OCCN, OCPN, and OC-DFG, compared

To illustrate how readability is affected by edge count, Figure 19 shows the control flow between the activities "Create Purchase Requisition," "Approve Purchase Requisition," "Delegate Purchase Requisition Approval," and "Create Request for Quotation." On the left (Figure 19a), the OCCN model (our tool) presents a clear, unidirectional flow. On the right (Figure 19b), the OCPN model (PM4Py) exhibits increased complexity due to the higher number of edges, making it difficult to follow the process flow as edges cross in multiple directions. Similarly, the OC-DFG model (Figure 19c) is cluttered with multiple edges connecting the same activities of the same object type, impairing readability. This results from the lack of support for choice and concurrency constructs, which requires multiple edges to represent parallelism.

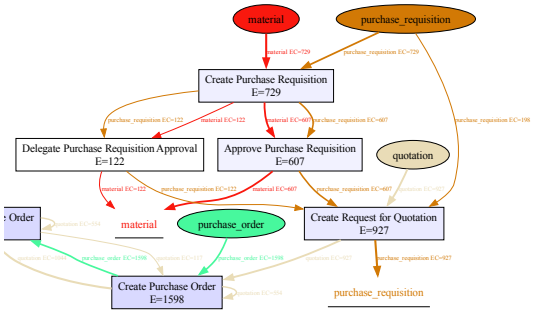
Finally, we calculated Edge Ratio metrics for each process log to quantify the degree of complexity increase in the edge dimension (arcs) across techniques. The 'Edge Ratio OCPN' represents the ratio between 'Edges OCPN' and 'Edges OCCN' (i.e., 'Edge Ratio OCPN' = 'Edges OCPN' / 'Edges OCCN'). Similarly, the 'Edge Ratio OC-DFG' represents the ratio between 'Edges OC-DFG' and 'Edges OCCN' (i.e., 'Edge Ratio OC-DFG' = 'Edges OC-DFG' / 'Edges OCCN'). These ratios provide a quantitative measure of model complexity in terms of edges. Figure 20 shows that, except for the CargoPickup process, the number of edges in OCPN and OC-DFG models is substantially greater than in OCCN.



(a) OCCN (our tool) – P2P process detail – edges



(b) OCPN (PM4Py) – P2P process detail – edges



(c) OC-DFG (PM4Py) – P2P process detail – edges

Figure 19. Model comparison: the impact of the number of edges on readability

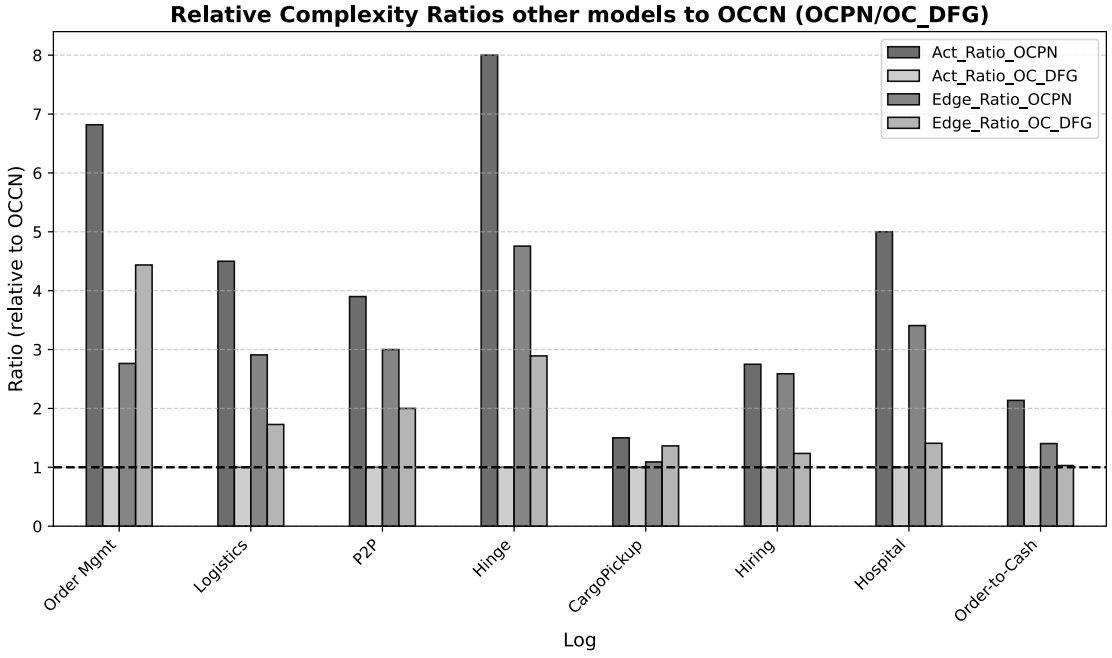


Figure 20. Edge Ratio and Activity Ratios in OCCN, OCPN, and OC-DFG, compared

In the same way, we calculated Activity Ratio metrics for each process log to quantify complexity in the activity dimension (nodes). The 'Act. Ratio OCPN' is defined as 'D.A. OCPN' / 'D.A. OCCN,' and the 'Act. Ratio OC-DFG' as 'D.A. OC-DFG' / 'D.A. OCCN.' Figure 20 shows that the 'Act. Ratio OCPN' is more pronounced than the 'Edge Ratio OCPN,' highlighting the impact of silent transitions on model simplicity. For OC-DFG models, the ratio is always 1.00 (Table 4), since they do not introduce additional activities, as previously noted.

Thus, both statistical and graphical evidence demonstrate that OCPN and OC-DFG models implemented in PM4Py require a greater number of entities to represent process control-flow behavior in OCEL 2.0 event logs than OCCN models in our approach. OCPN and OC-DFG result in cluttered models with reduced readability. In contrast, OCCN models implemented in our approach provide uncluttered representations that include only the activities found in the event log, with edges that clearly depict the control flow and support concurrency and choice.

6 Discussion

The implemented artifact is developed to discover OCCN from OCEL, making it compatible to be used along with other process mining libraries like PM4Py. It is particularly important as one can filter logs based on functionality provided by other libraries and discover relevant OCCN using our developed artifact. Despite the promising results and positive user evaluations, several limitations and potential threats to validity should be acknowledged to provide an appropriate context for the findings of this study.

Although the proposed method effectively discovers OCCN models with causality within individual object types, it does not account for inter-object type bindings, which are causal relationships that span across different object types. Consequently, patterns involving complex interactions between related objects, such as between 'orders' and 'items,' are not explicitly represented in the resulting OCCN models – even though users can implicitly follow them. This limits the method's ability to capture cross-object process dynamics fully. Addressing this limitation is a key direction for future work. A possible approach is the identification of cross-object bindings, while preserving model readability by filtering out infrequent ones and adding different edge types.

The user study was conducted with student participants. While many had relevant educational backgrounds and reported experience with other business process modeling languages, the generalizability of the results is limited. To strengthen external validity, future evaluations should include a larger and more diverse participant pool, particularly involving professionals from industry settings, through a qualitative model comparison analysis of OCCN, OCPN, and OC-DFG models. In addition, quantitative data can be collected using pattern recognition to assess understandability and TAM to measure user acceptance.

Regarding *future directions*, there is a need to extend the discovery algorithm to identify and model patterns that span across different object types. Currently, OCCN captures intra-object type behavior effectively; however, uncovering causal relationships that involve interactions between multiple object types could provide an even richer and more holistic understanding of complex processes. Exploring these inter-object type patterns remains an important step toward advancing the capabilities and applicability of OCPM.

Another promising direction is to explore how graph-based process mining [39] methods can apply graph simplification techniques to improve visualizations. While OCEL can be converted into event knowledge graphs [40] and vice versa [41], these graphs often become too complex for practical use. This challenge is especially evident with OCEL-based approaches, as related tools continue to evolve. As demonstrated in this article, graph simplification can enhance the user experience. Therefore, further research is needed to determine how these techniques can help end users interact more effectively with graph-based process mining.

We have demonstrated how edge-coarse-graining can enhance the process model by changing the level of abstraction to more suitable for end users. It is important to note that there are different ways to abstract a model, e.g., one can use OLAP techniques [23], [24]. However, these approaches demand the existence of domain knowledge to guide the dimensions under which such OLAP operations will be applied. Currently, there is no automatic way to define and apply them. Object-Centric Event Data can also be abstracted with the help of analysts and enriched to incorporate process copes for facilitating the discovery of more aspects [42]. As future work, it will be interesting to also study how a combination of these techniques can enhance process discovery algorithms.

7 Conclusion

In this article, we introduced a novel approach for discovering Object-Centric Causal Nets (OCCN), addressing key limitations of existing object-centric process discovery methods related to complexity, interpretability, and support for concurrency and choice. By extending traditional causal nets and incorporating an edge-coarse-graining technique, our method generates simplified yet expressive models that are easier for stakeholders to comprehend. We implemented the approach in Python and conducted a comparative evaluation against Object-Centric Petri Nets (OCPN) and Object-Centric Directly-Follows Graphs (OC-DFG). The results show that OCCN achieves higher user acceptance, provides stronger support for pattern recognition tasks, and produces simpler models with improved readability. These findings highlight the potential of OCCN as a user-centered alternative for object-centric process analysis.

References

[1] W. M. P. van der Aalst, "Data science in action," in *Process Mining: Data Science in Action*. Springer, 2016, pp. 3–23. Available: https://doi.org/10.1007/978-3-662-49851-4_1

[2] W. M. P. van der Aalst, "Object-centric process mining: Unraveling the fabric of real processes," *Mathematics*, vol. 11, no. 12, article 2691, 2023. Available: https://doi.org/10.3390/math11122691

[3] A. Berti, I. Koren, J. N. Adams, G. Park, B. Knopp, N. Graves, M. Rafiei, L. Liß, L. T. G. Unterberg, Y. Zhang *et al.*, "OCEL 2.0 specification," 2024, arXiv. Available: https://arxiv.org/abs/2403.01975

[4] W. M. P. van der Aalst and A. Berti, "Discovering object-centric petri nets," *Fundamenta Informaticae*, vol. 175, no. 1-4, pp. 1–40, 2020. Available: https://doi.org/10.3233/FI-2020-1946

[5] W. M. P. van der Aalst, "Object-centric process mining: Dealing with divergence and convergence in event data," in *Software Engineering and Formal Methods (SEFM 2019)*, vol. 11724, Springer, 2019, pp. 3–25. Available: https://doi.org/10.1007/978-3-030-30446-1_1

[6] S. Khayatbashi, V. Sjölind, A. Granåker, and A. Jalali, "AI-enhanced business process automation: A case study in the insurance domain using object-centric process mining," in *Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2025. Lecture Notes in Business Information Processing*, vol. 558, Springer, 2025, pp. 3–18. Available: https://doi.org/10.1007/978-3-031-95397-2_1

[7] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989. Available: https://doi.org/10.2307/249008

[8] E. de Moura Figueiredo and A. Jalali, "Discovering object-centric causal nets with edge-coarse-graining in process mining," in *Perspectives in Business Informatics Research, BIR 2025, Lecture Notes in Business Information Processing*, vol. 562. Springer, 2025, pp. 201–218. Available: https://doi.org/10.1007/978-3-032-04375-7_13

[9] W. M. P. van der Aalst, A. Adriansyah, and B. van Dongen, "Causal nets: A modeling language tailored towards process discovery," in *CONCUR 2011 – Concurrency Theory*, ser. Lecture Notes in Computer Science, vol. 6901. Springer, 2011, pp. 28–42. Available: https://doi.org/10.1007/978-3-642-23217-6_3

[10] A. Weijters, W. M. P. van der Aalst, and A. Alves De Medeiros, *Process mining with the Heuristics Miner algorithm*. Eindhoven: Technische Universiteit Eindhoven, 2006.

[11] E. de Moura Figueiredo and A. Jalali, “Object-centric process mining for public sector transformation,” in *Joint Proceedings of the BIR 2025 Workshops and Doctoral Consortium, co-located with 24th International Conference on Perspectives in Business Informatics Research (BIR 2025)*, vol. 4034, 2025, pp. 184–197. Available: https://ceur-ws.org/Vol-4034/paper87.pdf

[12] P. Johannesson and E. Perjons, *An Introduction to Design Science*. Springer, 2014. Available: https://doi.org/10.1007/978-3-319-10632-8

[13] W. M. P. Van Der Aalst, T. Weijters, and L. Maruster, “Workflow mining: discovering process models from event logs,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142, 2004. Available: https://doi.org/10.1109/TKDE.2004.47

[14] A. Weijters and W. M. P. Van Der Aalst, “Rediscovering workflow models from event-based data using little thumb,” *Integrated Computer-Aided Engineering*, vol. 10, no. 2, pp. 151–162, 2003. Available: https://www.medra.org/servlet/aliasResolver?alias=iospress&doi=10.3233/ICA-2003-10205

[15] A. Weijters and J. Ribeiro, “Flexible heuristics miner (FHM),” in *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE, 2011, pp. 310–317. Available: https://doi.org/10.1109/CIDM.2011.5949453

[16] F. Mannhardt, M. de Leoni, and H. Reijers, “Heuristic mining revamped: an interactive, data-aware, and conformance-aware miner: 15th international conference on business process management (BPM 2017),” in *Proceedings of the BPM Demo Track and BPM Dissertation Award*, CEUR Workshop Proceedings, vol. 1920, 2017, pp. 1–5. Available: http://ceur-ws.org/Vol-1920/BPM_2017_paper_167.pdf

[17] S. K. Vanden Broucke and J. De Weerdt, “Fodina: A robust and flexible heuristic process discovery technique,” *Decision Support Systems*, vol. 100, pp. 109–118, 2017. Available: https://doi.org/10.1016/j.dss.2017.04.005

[18] C. W. Günther and W. M. P. Van Der Aalst, “Fuzzy mining – adaptive process simplification based on multi-perspective metrics,” in *Business Process Management, BPM 2007. Lecture Notes in Computer Science*. Springer, 2007, vol. 4714, pp. 328–343. Available: https://doi.org/10.1007/978-3-540-75183-0_24

[19] E. De Moura Figueiredo, “Discovering object-centric causal nets by merging causal nets from independent object type analyses,” Master’s thesis, Stockholm University, 2024. Available: https://su.diva-portal.org/smash/record.jsf?pid=diva2%3A1955576&dswid=-246

[20] A. Jalali, “Object Type Clustering Using Markov Directly-Follow Multigraph in Object-Centric Process Mining,” *IEEE Access*, vol. 10, pp. 126 569–126 579, 2022. Available: https://doi.org/10.1109/access.2022.3226573

[21] C. Song, S. Havlin, and H. A. Makse, “Self-similarity of complex networks,” *Nature*, vol. 433, pp. 392–395, 2005. Available: https://doi.org/10.1038/nature03248

[22] D. Gfeller and P. De Los Rios, “Spectral coarse graining of complex networks,” *Physical Review Letters*, vol. 99, no. 3, p. 038701, 2007. Available: https://doi.org/10.1103/physrevlett.99.038701

[23] S. Khayatbashi, N. Miri, and A. Jalali, “OLAP operations for object-centric process mining,” in *Intelligent Information Systems. CAiSE 2025. Lecture Notes in Business Information Processing*, vol. 557. Springer, 2025, pp. 111–118. Available: https://doi.org/10.1007/978-3-031-94590-8_14

[24] S. Khayatbashi, N. Miri, and A. Jalali, “Advancing object-centric process mining with multi-dimensional data operations,” 2025, arXiv. Available: https://arxiv.org/abs/2412.00393

[25] N. Miri and A. Jalali, “Uncovering patterns in object-centric process mining: An approach using drill-down and roll-up techniques,” in *Information Integration and Web Intelligence*, vol. 15343. Springer, 2025, pp. 49–54. Available: https://doi.org/10.1007/978-3-031-78093-6_4

[26] N. Miri, S. Khayatbashi, J. Zdravkovic, and A. Jalali, “OCPM^2: Extending the process mining methodology for object-centric event data extraction,” in *Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2025. Lecture Notes in Business Information Processing*, vol. 558. Springer, 2025, pp. 123–140. Available: https://doi.org/10.1007/978-3-031-95397-2_8

[27] X. Lu, “Causal-net category,” 2022, version Number: 5, 2022. Available: https://arxiv.org/abs/2201.08963

[28] B. Knopp and W. M. P. van der Aalst, “Order management object-centric event log in OCEL 2.0 standard,” 2023, Zenodo. Available: https://doi.org/10.5281/zenodo.8337463

[29] B. Knopp and N. Graves, “Container logistics object-centric event log [data set],” 2023, Zenodo. Available: https://doi.org/10.5281/zenodo.8428084

[30] G. Park and L. T. genannt Unterberg, “Procure-to-payment (P2P) object-centric event log in OCEL 2.0 standard,” 2023, Zenodo. Available: https://doi.org/10.5281/zenodo.8412920

[31] M. Heinisch, N. Graves, and W. M. P. van der Aalst, "sOCEL 2.0: A sustainability-enriched OCEL of a hinge production process (1.0) [data set]," 2024, Zenodo. Available: https://doi.org/10.5281/zenodo.13638681

[32] J. Wei, C. Ouyang, W. Ma, D. Jiang, J. Xia, A. ter Hofstede, Y. Wang, and L. Huang, "From conventional to IoT-enhanced: Simulated object-centric event logs for real-life logistics processes," in *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration and Resources Forum at BPM 2024*, 2024, vol. 3758, pp. 106–110. Available: https://ceur-ws.org/Vol-3758/paper-19.pdf

[33] A. Berti, "Simulated object-centric event logs (OCEL 2.0) for order-to-cash, procure-to-pay, hiring, and hospital patient lifecycle processes [data set]," 2024, Zenodo. Available: https://doi.org/10.5281/zenodo.13879980

[34] A. Jalali, "Evaluating user acceptance of knowledge-intensive business process modeling languages," *Software and Systems Modeling*, vol. 22, no. 6, pp. 1803–1826, 2023. Available: https://doi.org/10.1007/s10270-023-01120-6

[35] A. Jalali, "Evaluating perceived usefulness and ease of use of cmmn and dcr," in *Enterprise, Business-Process and Information Systems Modeling. BPMDS EMMSAD 2021. Lecture Notes in Business Information Processing*, vol. 421. Springer, 2021, pp. 147–162. Available: https://doi.org/10.1007/978-3-030-79186-5_10

[36] A. Jalali, F. M. Maggi, and H. A. Reijers, "A hybrid approach for aspect-oriented business process modeling," *Journal of Software: Evolution and Process*, vol. 30, no. 8, p. e1931, 2018. Available: https://doi.org/10.1002/smr.1931

[37] A. Jalali, "Weaving of aspects in business process management," *Complex Systems Informatics and Modeling Quarterly*, no. 15, pp. 24–44, 2018. Available: https://doi.org/10.7250/csimq.2018-15.02

[38] J. Davies, "Word cloud generator," Available: https://www.jasondavies.com/wordcloud.

[39] A. Jalali, "Graph-based process mining," in *Process Mining Workshops. ICPM 2020. Lecture Notes in Business Information Processing*, vol. 406. Springer, 2020, pp. 273–285. Available: https://doi.org/10.1007/978-3-030-72693-5_21

[40] S. Khayatbashi, O. Hartig, and A. Jalali, "Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis," in *Conceptual Modeling. ER 2023. Lecture Notes in Computer Science*, vol. 14320. Springer, 2023, pp. 220–238. Available: https://doi.org/10.1007/978-3-031-47262-6_12

[41] S. Khayatbashi, O. Hartig, and A. Jalali, "Transforming object-centric event logs to temporal event knowledge graphs," in *Business Process Management Workshops. BPM 2024. Lecture Notes in Business Information Processing*, vol. 534. Springer, 2024, pp. 300–313. Available: https://doi.org/10.1007/978-3-031-78666-2_23

[42] S. Khayatbashi, M. Rafiei, J. Chen, T. Kampik, G. Berg, and A. Jalali, "Enriching object-centric event data with process scopes: A framework for aggregation and analysis," in *International Conference on Process Mining*, 2025.