A Framework for Advanced Case Notions in Object-Centric Process Mining

Jan Niklas van Detten[1,2] ✉, Pol Schumacher[2], and Sander J.J. Leemans[1,3]

[1] RWTH University, Aachen, Germany
[2] Celonis, Munich, Germany
[3] Fraunhofer FIT, Aachen, Germany

Abstract. Real-life processes involve interacting business objects of different types. Object-centric event logs capture the execution of activities in such processes. An important step in the analysis of such logs is the identification of sets of objects which characterize an execution of the process, called a case. Given a case notion, visualizations can be constructed to display the relations between the executed activities and the involved business objects. Depending on the utilized case notion, these visualizations can quickly become excessively complex, impeding human analysis, or may oversimplify the underlying process, inducing flawed insights. To combat these issues, new case notions are needed to reduce complexity while representing relevant structures of the underlying business process correctly. In this paper, we propose continuous measures to quantify how correctly an object-centric case notion adheres to a given log and how complex the resulting visualizations are. These measures allow us to conceptualize the search for new object-centric case notions as a joint optimization problem among the two quality dimensions of correctness and simplicity. As a result, we can provide a new case notion that significantly reduces complexity in comparison to existing techniques, while preserving relevant object interactions. To evaluate our approach, we apply it to a range of real-life logs and find that major complexity reductions can be achieved without causing excessive correctness issues.

Keywords: process mining · object-centric · case notions · visualizations

1 Introduction

The research area of process mining provides techniques to analyze business processes based on digital execution track records, called event logs. These logs contain sequences of events with the labels of executed activities. For analytical purposes, it is important to visualize the relation between these activities.

Traditionally, process mining techniques have been used to study processes in isolation. For this purpose, all events associated to one particular object, like a loan application for example, are considered an execution of the process, called a case. However, real-life business processes rarely exist in such a single-object vacuum. Instead, processes often involve multiple business objects that might

interact in shared events. A loan application might, for example, lead to the creation of a loan contract and require multiple payments to be issued.

Object-centric process mining techniques address this reality by considering anything that participates in a process as an object. Object-centric event logs are sequences of events with activities associated to sets of objects. To study the interactions between objects, the traditional single-object case notion is not sufficient. The events associated to a payment might, for example, be influenced by the events of the preceding loan application. Therefore, it would be insufficient to exclusively focus on a single object. Instead, a set of objects with a set of associated events characterizes an execution, i.e. a case, of such processes.

The division of object-centric event logs into cases of associated objects and event sets can induce unintended side effects when done incorrectly, such as duplicated events, lost events or incorrect relationships between them [1]. Existing work to guarantee the absence of such phenomena has been proposed in [3]. However, these techniques do not take the effect of the selected case notion on the complexity of corresponding visualizations into account. As a result, practitioners are currently faced with the choice between two extreme options. That is, they either use the traditional case notion, which may induce severe correctness issues in the object-centric setting, or apply the approach from [3], which produces only few, large cases for processes with many object interactions.

In this paper, we propose an evaluation framework to conceptualize the simplicity and correctness of object-centric case notions as continuous quality measures. Additionally, we introduce a new object-centric case notion that significantly reduces the complexity of case visualizations, while preserving their ability to represent large parts of the underlying business process correctly. We apply our approach to a range of real-life logs and find that major complexity reductions can be achieved without causing excessive correctness issues.

2 Preliminaries

In this section, we summarize required background information on object-centric event logs, existing case notions and common correctness criteria for them.

2.1 Object-Centric Event Logs

An object-centric event log is an ordered sequence of events $\langle (a_1, O_1), \ldots, (a_n, O_n) \rangle$ with activities a_i and sets of objects O_i. We write Σ, Θ and Ω for all activities, objects and object types in such a log respectively. Each object is associated to its object type with the injective type function $\omega : \Theta \mapsto \Omega$. An object set O can be projected onto the subset of objects with the type $ot \in \Omega$ with $O \restriction_{ot}$.

Table 1 shows an example log of a loan application process. It involves applications (`a`), loans (`l`) and payments (`p`) in addition to three employees (`e`) and a software system (`s`). Applications are submitted to be checked by an employee, which can reject it or grant a loan based upon it. Associated payments are subsequently payed, with a second employee checking the correctness of each payment. The system tracks which applications are submitted, rejected and granted.

Table 1. Object-centric event log of a loan application process.

⟨SUBMIT $\{\mathtt{a}_1, \mathtt{s}_1\}$,	CHECK $\{\mathtt{a}_1, \mathtt{e}_1\}$,	SUBMIT $\{\mathtt{a}_2, \mathtt{s}_1\}$,	DENY $\{\mathtt{a}_1, \mathtt{e}_1, \mathtt{s}_1\}$,
SUBMIT $\{\mathtt{a}_3, \mathtt{s}_1\}$,	CHECK $\{\mathtt{a}_2, \mathtt{e}_1\}$,	GRANT $\{\mathtt{a}_2, \mathtt{e}_1, \mathtt{s}_1, \mathtt{l}_1\}$,	CHECK $\{\mathtt{a}_3, \mathtt{e}_1\}$,
PAY $\{\mathtt{l}_1, \mathtt{p}_1, \mathtt{e}_1, \mathtt{e}_2\}$,	DENY $\{\mathtt{a}_3, \mathtt{e}_1, \mathtt{s}_1, \}$,	SUBMIT $\{\mathtt{a}_4, \mathtt{s}_1, \}$	PAY $\{\mathtt{l}_1, \mathtt{p}_2, \mathtt{e}_1, \mathtt{e}_2\}$,
CHECK $\{\mathtt{a}_4, \mathtt{e}_1\}$,	GRANT $\{\mathtt{a}_3, \mathtt{e}_1, \mathtt{s}_1, \mathtt{l}_2\}$,	PAY $\{\mathtt{l}_2, \mathtt{p}_3, \mathtt{e}_1, \mathtt{e}_3\}$	PAY $\{\mathtt{l}_2, \mathtt{p}_4, \mathtt{e}_1, \mathtt{e}_3\}\rangle$

An object-centric event log can be interpreted as an undirected bipartite graph structure. For this purpose, every object and event is conceptualized as a node. An arc between them indicates that the object is involved in the event.

We refer to such a graph as a log graph for a given object-centric event log $L = \langle (a_1, O_1), \ldots, (a_n, O_n) \rangle$. Formally, a log graph $L_G = (E, O, A)$ consists of event nodes $E = \{\bullet_i \mid 1 \leq i \leq n\}$ and object nodes $O = \{\bullet_o \mid o \in \Theta\}$, which are connected with the undirected arcs $A = \{(\bullet_i, \bullet_o) \mid o \in O_i\}$. Figure 1 shows the log graph for our example log from Table 1, with colors indicating object types.

2.2 Object-Centric Case Notions

A case notions projects object-centric event logs onto associated event and sets of objects. Each of these object-event combinations represents an independent execution of the process. In this paper, we consider this a division operation on the log graph of a given object-centric event log.

Formally, an object centric case notion is a projection of a given log graph on a set of subgraphs. For a given log L with the log graph $L_G = (E, O, A)$, a case notion c is any method that projects L_G on a set of subgraphs, i.e. $c : (E, O, A) \mapsto \mathbb{P}(\{E', O', A' \mid E' \subseteq E \wedge O' \subseteq O \wedge A' = A \cap (E' \times O')\})$. Note

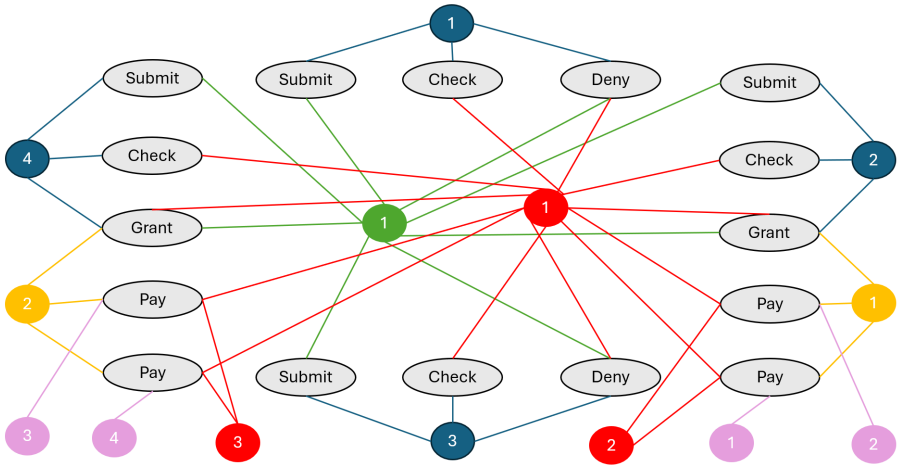

Fig. 1. Visualization of the log graph for the example log in Table 3 with applications (blue), loans (yellow), payments (violet), employees (red) and systems (green).

that such a case notion, without further restrictions, might duplicate or drop event or object nodes of the original graph and the edges between them.

2.3 Traditional Case Notion

The traditional case notion evolves around individual objects of the same type and their respective events. Given a log graph and an object type, each case only has a single object of this type and all the event nodes connected to it.

Formally, the results of the traditional case notion c_{ot} for an object type $ot \in \Omega$ and the log graph $L_G = (E, O, A)$ are given by the subgraphs $c_{ot} = \{(E', O', A') \mid O' = \{\bullet_o\} \wedge \omega(o) = ot \wedge E' = \{\bullet_e \mid \exists(\bullet_o, \bullet_e) \in A\} \wedge A' = O' \times E'\}$.

Note that the results of the traditional case notion heavily depend on the selected object type. For the example log graph in Figure 1 and the object types of the involved employees, the resulting cases can be seen in Figure 2

The usage of the traditional case notion can cause three correctness issues, known as divergence, convergence and deficiency [1]. Convergence and deficiency refer to events being associated to multiple cases or no case at all. Divergence describes the assignment of events to the same case without including the involved objects in which they differ. In our example we can observe convergence, since the payment events are duplicated among the two employees. Deficiency occurs, because the submission events are not related to any employee. Divergence occurs since the events of different applications involve the same employee.

For the scope of this paper, we formally consider divergence, convergence and deficiency at the level of activities and object types. That is, we speak of convergence of $ot \in \Omega$ and $a \in \Sigma$ if $\exists(a, O) \in L$ with $|O\lfloor_{ot}| > 1$. Similarly, we refer to deficiency of $ot \in \Omega$ and $a \in \Sigma$ if $\exists(a, O) \in L$ with $|O\lfloor_{ot}| < 1$. Divergence occurs for a and ot if $\exists(a, O), (a, O') \in L \colon O \neq O' \wedge (O \cap O')\lfloor_{ot} \neq \emptyset$.

2.4 Connected Component Case Notion

The correctness issues of divergence, convergence and deficiency in object-centric case notion have been addressed in literature. The approach in [3] proposes to

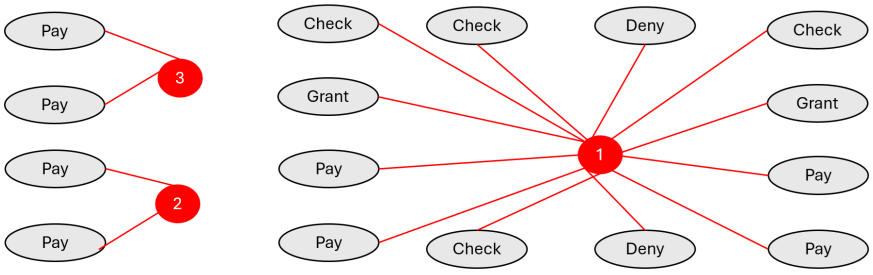

Fig. 2. Visualization of the cases resulting from the application of the traditional case notion on the log graph in Figure 1 for the object types of employees (red).

use each connected component in the log graph of an object-centric event log as a case. The approach proves that this strategy is the only one that can guarantee the absence of correctness issues due to divergence, convergence and deficiency.

For our example log graph in Figure 1, the connected component case notion results in a single case that contains the entire log graph. The reason for this phenomenon are highly connected objects, such as the employee and the system.

3 Optimization Framework

In this section, we present our evaluation framework for object-centric case notions. First, we propose continuous measures for the correctness and simplicity of a given case notion in Section 3.1 and 3.2 respectively. Then, we sketch how these measures can be used to conceptualize new object-centric case notions as an optimization problem across the two quality dimensions in Section 3.3.

3.1 Graph-Based Correctness Measures

Our correctness criteria for object-centric case notions focus on the correct and lossless division of objects, events and the relations between them. With regards to the graph-based interpretation, these criteria correspond to the required absence of duplicated or missing nodes and edges of the log graph. Our measures are backwards compatible to existing correctness notions, i.e a case notion with perfect scores is free of any issues due to divergence, convergence and deficiency.

Formally, for a given case notion c and log graph $L_G = (E, O, A)$, an object node $\bullet_o \in O$ adheres to the correctness criteria of the case notion, if it appears in exactly one case, i.e. $|\{(E', O', A') \in c(L_G) \mid \bullet_o \in O'\}| = 1$. We write O_c for all objects nodes that adhere to the case notion. Similarly, we define the adherence of an event node $\bullet_e \in E$ to the correctness criteria of a case notion c with $|\{(E', O', A') \in c(L_G) \mid \bullet_e \in E'\}| = 1$. We write E_c for the set of adhering event nodes. Analogously, we define the set of adhering edges of the log graph for the case notion c with $A_c = \{a \in A \mid |\{(E', O', A') \in c(L_G) \mid a \in A'\}| = 1\}$.

Based upon the notion of adhering nodes and edges, we define the correctness measures for a given log graph and case notion as the average value of $|O_c| \cdot |O|^{-1}$, $|E_c| \cdot |E|^{-1}$ and $|A_c| \cdot |A|^{-1}$. More sophisticated aggregation methods will be investigated in future work. Table 2 shows the correctness measures for the graph in Figure 1 and the case notions explained in Section 2.3 and Section 2.4.

Table 2. Quality measures for the log graph from Figure 1.

Notion	$\lvert O_c\rvert \cdot \lvert O\rvert^{-1}$	$\lvert E_c\rvert \cdot \lvert E\rvert^{-1}$	$\lvert A_c\rvert \cdot \lvert A\rvert^{-1}$	Correctness	Simplicity	Total
$c_{employee}$	0.21	0.50	0.35	0.35	0.79	0.49
$c_{payment}$	0.29	0.25	0.09	0.20	0.93	0.34
c_{system}	0.07	0.50	0.17	0.25	0.70	0.37
c_{loan}	0.14	0.38	0.13	0.22	0.87	0.35
$c_{application}$	0.29	0.75	0.26	0.43	0.87	0.58
c_{cc}	1.00	1.00	1.00	1.00	0.00	0.00

3.2 Graph-Based Simplicity Measures

The complexity of case visualizations is strongly connected to the number of objects and events per case. Therefore, we use the average number of nodes in each case identified by a given case notion as a proxy measure for the complexity of the corresponding visualizations. To again have a measure on a scale from zero to one, we set that number into relation with the overall size of the log graph.

Formally, for a given log graph $L_G = (E, O, A)$ and case notion c, we first determine the average case size with $S(L_G, c) = \sum_{(E',O',A') \in c(L_G)} (|E'| + |O'|) \cdot |c(L_G)|^{-1}$. Based upon the resulting value, we subsequently define the simplicity measure for the case notion and the log graph as $1 - S(L_G, c) \cdot (|E| + |O|)^{-1}$. Table 2 shows the simplicity of existing case notions on the log graph from Figure 2.

3.3 Graph Based Optimization

Given a set of potential case notions, we can use the measures proposed in the previous sections to select the case notion that hits the best trade off between correctness and simplicity. For this purpose, we need to combine the two quality dimensions into a single, overall score. For the scope of this paper, we utilize the harmonic mean between the correctness and simplicity for this purpose, similar to the commonly used F1-score in process mining.

For the existing case notions with the corresponding measures in Table 2, we can use this accumulation to decide which of those is best to use for the example log graph in Figure 1. However, note that even the best case notion, which is the traditional case notion for applications, only achieves 0.58 as a total score.

4 Object-Centric Case Notions

In this section we motivate and formalize a new case notion to optimize the balance between simplicity and correctness of object-centric process visualizations

4.1 Resource-Like Business Objects

Considering the correctness and simplicity of existing case notions in Table 2, two patterns become apparent. The traditional case notions for individual object types are simple, but ignore many objects and events that refer to different object types. In contrast, the connected component case notion is correct, but excessively complex, due to some highly connected objects in the log graph. These objects are often of resource-like nature, interacting with many objects that would traditionally be considered as individual cases. Therefore, we propose a new object-centric case notion, that determines the connected components in a log graph but does not consider connections that arise from resource-like objects.

Algorithm 1 Advanced Object-Centric Case Notion

function $c'_{ot}((E, O, A), div)$
  $result \leftarrow \{\}$
  for $\bullet_o \in \{\bullet_o \in O \mid \omega(o) = ot\}$ **do**
    $O', O'', E', E'' \leftarrow \{\bullet_o\}, \{\bullet_o\}, \{\}, \{\}$
    while $O'' \neq \emptyset \vee E'' \neq \emptyset$ **do**
      $E'' \leftarrow \{\bullet_e \in E \setminus E' \mid \exists \bullet_o \in O'' : (\bullet_e, \bullet_o) \in A\}$
      $O'' \leftarrow \{\bullet_o \in O \setminus O' \mid \exists \bullet_e \in E'' : (\bullet_e, \bullet_o) \in A \wedge \omega(o) \notin div(\bullet_e) \cup \{ot\}\}$
      $O''' \leftarrow \{\bullet_o \in O \setminus O' \mid \exists \bullet_e \in E'' : (\bullet_e, \bullet_o) \in A \wedge \omega(o) \in div(\bullet_e) \setminus \{ot\}\}$
      $E', O' \leftarrow E' \cup E'', O' \cup O'' \cup O'''$
    $result \leftarrow result \cup \{(E', O', A')\}$
  return result

4.2 Advanced Case Notion

Similar to the traditional case notion, we use a given object type as a starting point for the construction of the cases. For each individual object of this type, we add all related events to the same case. Then we utilize the idea of the connected components and repeatedly add objects and events to the case that are transitively related to the starting object. However, if the path to an object node leads through an event node with an activity on which this object's type diverges, we stop. As a result, connections caused by resource-like business objects are not followed when constructing a case. Algorithm 1 shows the code for our advanced case notion with a given start type. The auxiliary function *div* maps each event node to its diverging object types. After determining the advanced case notion for each object type, we again use our framework to pick the best one. Applying this strategy to our running example, results in a case notion with 0.80 as the total score. Figure 3 shows the resulting cases for our example.

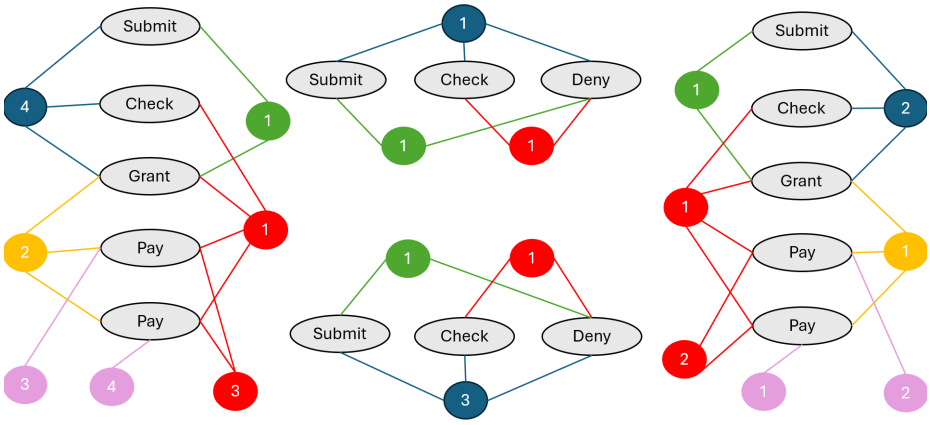

Fig. 3. Advanced case notion with applications as the starting object type.

5 Evaluation

In this section, we evaluate the technique proposed in this paper. For this purpose, we prototypically implemented our approach in Python. We utilize a large range of synthetically generated and real-life object-centric event logs that are publicly available and shown in Table 3. We apply our approach to these logs and observe the runtime required to apply our framework to evaluate its computational feasibility. Additionally, we investigate how well our proposed case notion from Section 3.3 performs quantitatively in comparison with the existing notions from Section 2.4 and Section 2.3. Lastly, we perform a qualitative comparison by manually investigating the visualizations of the different case notions.

5.1 Runtime Efficiency

To evaluate the computational feasibility of our framework, we measured the time needed to determine and evaluate the traditional, connected component and advanced case notion respectively. The traditional and advanced case notion iterate over all possible start object types to determine the best one. All case notions used the same data structure for the representation of the log graphs. All experiments were performed on a consumer grade laptop with an Intel-Core i5-8265U processor with exclusive access to 16GB of working memory. We set a runtime limit of one hour per object-centric event log and case notion.

The resulting runtime measurements can be seen in Table 3, illustrating the computational feasibility of our framework and the advanced case notion. Across all logs and case notions, we only observed two time outs. All of them occurred during the connected component based approach. Upon closer investigation, we found this was caused by large connected components in the corresponding log

Table 3. Statistics of utilized object-centric logs and measured framework runtime for the traditional (TD), connected component (CC) and advanced (AD) case notion.

Log	Objects	Types	Events	Activities	TD (s)	CC (s)	AD (s)
BPIC15$_1$ [8]	1269	4	52217	289	11	9	46
BPIC15$_2$ [8]	859	4	44354	304	8	6	40
BPIC15$_3$ [8]	1465	4	59681	277	12	10	53
BPIC15$_4$ [8]	1084	4	47293	272	11	8	43
BPIC15$_5$ [8]	1202	4	59083	285	13	10	55
BPIC17 [8]	106162	4	1202267	26	140	t/o	451
BPIC19 [8]	330685	4	1595923	42	336	t/o	2171
Github [10]	28317	2	27842	67	68	87	1941
O2C [4, 10]	107767	19	98350	23	50	208	105
P2P [4, 10]	74489	8	24854	32	11	27	44
HR [10]	1505	6	6980	16	2	1	5
Logistics [10]	11521	5	22367	11	10	18	77
Transfers [10]	2500	5	10319	3	3	2	7
MIMIC [7]	3007	3	13410	3	2	3	4

graphs. While this is not a problem for the detection of connected components, it increases the runtime for the implementation of our measures drastically. The traditional and advanced case notion did not suffer from the same issue, having maximal runtime values of 6 and 37 minutes respectively. As expected, the traditional case notion outperformed the advanced case notion on all logs, since it only needs to consider object nodes of a single type, leading to smaller cases.

5.2 Case Notion Performance

Next, we investigate how well the traditional, connected component and advanced case notions perform on the logs from Table 3. For this purpose we measure simplicity, correctness and overall score as specified in Section 3.

The resulting scores can be seen in Table 4, confirming the pattern motivated in our running example. For most logs, the traditional case notion results in almost perfect simplicity scores, but suffers from correctness issues. Reversely, the connected component notion guarantees perfect correctness, but suffers from poor simplicity. However, the advanced notion successfully balances the two quality dimensions, resulting in the best overall score for 12 out of 14 logs.

Two interesting outliers are the two logs describing an order-to-cash and procure-to-pay process, that were extracted from simulated SAP instances [4]. For both of these logs, the connected component approach achieves the best score, with a significant distance to our proposed case notion. Upon closer investigation of these two logs, we found that these logs contain events that only have objects of diverging object types associated to them. As a result, our case notion is not able to reach those events from, otherwise optimal, starting types.

Table 4. Measured scores for the simplicity and correctness of the traditional (TD), connected component (CC) and advanced (AD) case notion respectively.

	Simplicity			Correctness			Total		
Log	TD	CC	AD	TD	CC	AD	TD	CC	AD
BPIC15$_1$ [8]	0.99	0.00	0.95	0.73	1.00	0.98	0.84	0.00	0.97
BPIC15$_2$ [8]	0.99	0.00	0.85	0.73	1.00	0.99	0.85	0.00	0.92
BPIC15$_3$ [8]	0.99	0.00	0.99	0.73	1.00	0.99	0.84	0.00	0.99
BPIC15$_4$ [8]	0.99	0.00	0.99	0.74	1.00	0.99	0.85	0.00	0.99
BPIC15$_5$ [8]	0.99	0.00	0.99	0.73	1.00	0.98	0.84	0.00	0.99
BPIC17 [8]	0.99	0.00	0.99	0.50	1.00	0.53	0.66	0.00	0.69
BPIC19 [8]	0.99	0.00	0.99	0.67	1.00	0.99	0.80	0.00	0.99
Github [10]	0.99	0.00	0.81	0.47	1.00	0.95	0.64	0.00	0.87
O2C [4, 10]	0.99	0.99	0.99	0.26	1.00	0.39	0.41	0.99	0.56
P2P [4, 10]	0.99	0.99	0.99	0.44	1.00	0.80	0.61	0.99	0.89
HR [10]	0.99	0.00	0.99	0.61	1.00	0.71	0.76	0.00	0.83
Logistics [10]	0.99	0.00	0.98	0.53	1.00	0.99	0.69	0.00	0.99
Transfers [10]	0.99	0.8	0.99	0.58	1.00	0.94	0.74	0.88	0.96
MIMIC [7]	0.99	0.5	0.99	0.76	1.00	0.99	0.86	0.66	0.99

5.3 Process Visualizations

Lastly, we manually inspect an example case visualization. Figure 4 shows a case in the human resource log, identified by our advanced case notion. We cannot visualize the corresponding case of the connected component approach, since it only identified a single case in the entire log, with more than eight thousand nodes. Using the corresponding cases of the traditional case notion for isolated object type induces several correctness issues. From the perspective of individual recruiters, it is not possible to deduct that they are assigned to applications in groups of three. For isolated applications, one can not deduct that multiple applications are submitted by the same applicant. Reversely, one cannot differentiate between events of independent applications from the applicants perspective. And by only considering a vacancy, a manager, or an offer, large parts of the process become invisible. Our advanced case notion manages to avoid these problems, without the excessive complexity of the connected components. The only correctness issue caused in exchange, is that the resource-like business objects of the managers, vacancies and recruiters are involved in multiple cases simultaneously.

5.4 Discussion

Our evaluation shows, that our framework for evaluating object-centric case notions is generally feasible in terms of runtime. Additionally, we found our proposed case notion to quantitatively outperform existing ones in terms of balancing simplicity and correctness. Our manual investigation example showcased that our approach offers an object-centric view on the executions of the process, without leading to excessively complex visualizations in return.

Limitations of our approach are the assumed structure of the input logs and the bias introduced by our measures. We rely on events being associated to at least one object of a non-diverging type. Approaches in literature exists to take

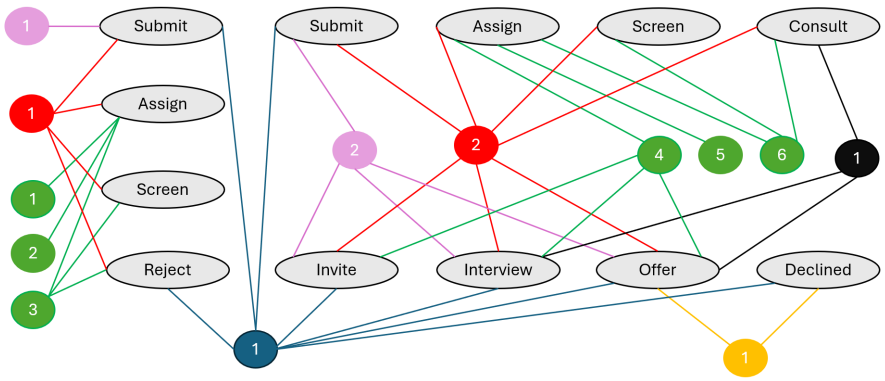

Fig. 4. Example case of the human resources log with applications (red), applicants (blue), recruiters (green), managers (black), vacancies (pink) and offers (yellow).

care of situations where this is not the case, in particular by using the notion of silent objects [5] or by applying artifact-centric techniques [11]. Incorporating these approaches as a preprocessing step into our case notion might solve the issues observed on the two SAP logs, which should be subject to further research.

While our measures for correctness are well rooted in existing correctness criteria for object-centric case notions, our simplicity measure induces a selection bias. Since it measures simplicity in relation to the overall size of the log graph, large cases can still occur as long as they are small in comparison to the overall graph. Our framework could be extended with penalty measures for the absolute case size, which would however sacrifice an unknown extent of correctness.

6 Related Work

In this section, we summarize related work on object-centric case notions and their correctness. The notion of correctness criteria associated to the absence of unique case identifiers in the object-centric setting has been discussed in [1] and addressed in [2]. The key properties of divergence, convergence and deficiency are reflected in our work as well, in the sense that perfect correctness scores imply the absence of these three issues. The work in [3] proves that such guarantees can only be given if, and only if, the case notion corresponds to the connected components in the underlying log graph. Our work extents these criteria by considering them on a continuous scale to evaluate the extent to which a case notion is correct, instead of treating correctness as a binary property. Additionally, [3] explores graph isomorphism techniques to evaluate if two cases are structurally the same, i.e. represent the same variant of the underlying process. The same techniques can be applied to every case notion that follows the graph based interpretation as described in Section 2 and hence also to our work. The problem of excessive case complexity has so far only addressed by resorting to preprocessing steps that remove objects of highly connected object types [3]. To the best of our knowledge, our approach is the first to conceptualize object-centric case notions as a continuous optimization problem among multiple quality dimensions.

Further ideas on graph-based interpretations of object-centric event logs have been proposed in [6]. The approach has been extended with additional functionality and implementations in [9,12] and utilizes the concepts of event knowledge graphs. In contrast to our conceptualization, which only considers edges between objects and events, event knowledge graph may contain further edges of different types. These are used to represent the time-based order between events from the perspective of different objects. Additionally, relations between objects can be expressed. Our work can be applied to these graphs as well, but the quality measures should be adjusted to also account for these additional edge types.

7 Conclusion

In this paper, we introduced a novel framework to continuously evaluate object-centric case notions among the quality dimensions of correctness and simplicity.

Based upon this framework, we introduced a new object-centric case notion which balances the two quality dimensions by enforcing a selection criteria on transitive relations between objects. As a result, we can construct relatively simple cases that still showcase important transitive relations between objects. We evaluated our approach threefold on a range of synthetic and real-life object-centric event logs. We found our approach to be feasible in terms of runtime, even for large input logs. Our proposed case notion quantitatively outperformed existing ones in terms of balancing simplicity and correctness. Additionally, we manually investigated some of the resulting process visualizations, exemplary confirming its qualitative advantages. Future work will be focused on providing a preprocessing pipeline to address the limitations caused by the assumptions on the input log structure, as discussed in Section 5.4. Additionally, further investigations should focus on the selection bias induced by our simplicity measures.

References

1. 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. Lecture Notes in Computer Science, vol. 11724, pp. 3–25. Springer (2019). https://doi.org/10.1007/978-3-030-30446-1_1
2. Adams, J.N., van der Aalst, W.M.P.: Addressing convergence, divergence, and deficiency issues. In: Business Process Management Workshops - BPM 2023 International Workshops. Lecture Notes in Business Information Processing, vol. 492, pp. 496–507. Springer (2023). https://doi.org/10.1007/978-3-031-50974-2_37
3. Adams, J.N., Schuster, D., Schmitz, S., Schuh, G., van der Aalst, W.M.P.: Defining cases and variants for object-centric event data. In: 4th International Conference on Process Mining, ICPM 2022, Bolzano, Italy, October 23-28, 2022. IEEE (2022)
4. Berti, A., Park, G., Rafiei, M., van der Aalst, W.M.P.: An event data extraction approach from SAP ERP for process mining. In: Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven. vol. 433. Springer (2021)
5. Jan Niklas van Detten, P.S., Leemans, S.J.: Object synchronizations and specializations with silent objects in object-centric petri nets. BPM, Proceedings (2024)
6. Fahland, D.: Multi-dimensional process analysis. In: Di Ciccio, C., Dijkman, R., del Río Ortega, A., Rinderle-Ma, S. (eds.) Business Process Management. pp. 27–33. Springer International Publishing, Cham (2022)
7. Johnson, A., Pollard, T., Shen, L., Lehman, L.w., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L., Mark, R.: Mimic-iii, a freely accessible critical care database. Scientific Data **3**, 160035 (05 2016)
8. Khayatbashi, S., Hartig, O., Jalali, A.: Bpi challenge 2015 -2019 (ocel) (2023)
9. Khayatbashi, S., Hartig, O., Jalali, A.: Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In: Conceptual Modeling. Springer Nature Switzerland, Cham (2023)
10. Koren, I., Adams, N., Berti, A.: OCEL 2.0 resources - www.ocel-standard.org. CoRR **abs/2403.01982** (2024)
11. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Cooperative Inf. Syst. **24**(1), 1550001:1–1550001:44 (2015)
12. Swevels, A., Fahland, D., Montali, M.: Implementing object-centric event data models in event knowledge graphs. In: Process Mining Workshops. pp. 431–443. Springer Nature Switzerland, Cham (2024)