Root Cause Analysis Using Rule Mining on Object-Centric Event Logs

Benedikt Knopp(✉), Mahsa Pourbafrani, and Wil van der Aalst

Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany
{knopp,mahsa.bafrani,wvdaalst}@pads.rwt-aachen.de

Abstract. In business processes, the behavior, evolution and interactions of objects influence the outcome of process instances, and thus the value that a business user may assign to them. For example, in an order-to-cash process, a complete and timely delivery of a package is desirable, but depends on what happens to other objects upstream, like production batches. Negative outcomes call for a Root Cause Analysis (RCA) on the process. While many approaches for RCA using process mining exist, none is native to object-centric frameworks and thus suitable for capturing dependencies across object types. This work presents a method for RCA that operates on object-centric event logs (OCELs). Given an OCEL, our method returns a set of association rules on the activity level. These rules associate descriptive patterns over the various object types occurring at events with patterns indicating the process outcome. The patterns are abstracted from the log with the help of a first-order logic based query engine. A case study confirmed that our method can identify problematic interactions across various object types in real-life business processes.

Keywords: Root Cause Analysis · Association Rule Mining · Object-Centric Process Mining · Process Querying

1 Introduction

A common goal in analyzing business processes is to understand operational problems. This endeavour is called *Root Cause Analysis (RCA)*. For the purpose of RCA, process mining techniques have been successfully deployed [1–3]. Existing approaches usually operate on event data that uses a fixed case notion, that is, logs in which each process instance relates to a unique object of a fixed type (e.g., a sales order or a delivery). The nature of processes, however, is not so simple, because objects of various types and their interactions constitute the processes of an organization. Hence, existing methods for RCA in process mining have to either neglect information from foreign case notions or flatten these information into the chosen case notion. This may cause issues of data redundancy, additional effort in maintaining data integrity, or information loss through aggregation.

A. Delgado and T. Slaats (Eds.): ICPM 2024 Workshops, LNBIP 533, pp. 57–69, 2025.
https://doi.org/10.1007/978-3-031-82225-4_5

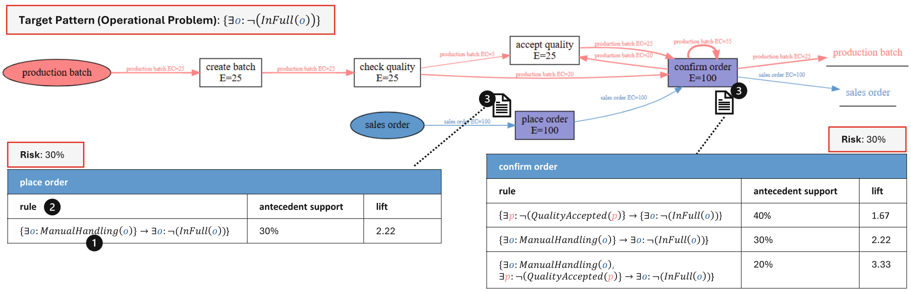

Fig. 1. Our method returns association rules on the event type level (3) that can be embedded into an object-centric process model. These rules (2) indicate which patterns are problematic with regards to an undesired process outcome. Patterns, in turn, are formulas over characteristics present at events, e.g., object characteristics (1).

To better account for the entangled nature of processes, object-centric process mining frameworks have been proposed [4,5]. The principle of these frameworks is to relate events not to a single case, but to arbitrarily many heterogeneously typed objects. While standard process mining utilities such as log standards [6] do exist for the object-centric setting, we are not aware of a designated RCA method. However, we argue that root causes of bad process outcomes may be found across interacting objects. For instance, in a company producing and selling goods, production of insufficient quality could be a cause for unfulfilled orders downstream. It is therefore the goal of the research presented here *(a)* to provide a method for RCA on object-centric process event logs and *(b)* to provide empirical evidence from real-life processes that the supposed root causes can in fact be found in object interactions, and that the method is able to discover this.

The output of our approach is illustrated in Fig. 1. Here, an exemplary order management process coupled to a production process is depicted as an object-centric process model [7]. On the one hand, *production batches* are created, undergo a quality check and are eventually released after accepting the quality. *Sales orders*, on the other hand, are placed by the customer and then confirmed by the company, at which point the order is assigned to supply from production units. In the process, it may happen that customer demands cannot be satisfied, causing the respective order to be not delivered *in full*. This is an operational problem that calls for an RCA. The process model is annotated with findings from applying our RCA approach, which is briefly described in the following.

We highlight three components that constitute our method, referring to the bullets in Fig. 1. *(1)* First, *patterns* are learned as descriptions of event characteristics. For this, a query language based on first-order logic is deployed in order to formulate these patterns across the objects occurring and interacting at the events. Since in object-centric processes, objects may occur at events in varying cardinalities, this query engine offers a means to precisely formulate properties across objects. *(2)* We mine for association rules [8] to identify among frequent combinations of patterns those that are likely to lead to the negative process outcome. This outcome is encoded in a *target pattern* on the object level. For example, here, the rule indicated at *(1)* expresses that if a newly placed order will be handled manually, the likelihood of an incomplete delivery is 2.22 times higher (*lift*), and 30% of the cases are handled manually (*antecedent support*). The third rule at *confirm order* exemplifies a rule that is a combination of patterns. *(3)* We suggest to embed mined rules into an object-centric process model [7]. This provides a map based on which business users could identify risky interactions and counteract as early as possible, given that rules provide actionable insights. To enable interpreting the output, we report on the model, as in Fig. 1, event counts and object flow counts. Also, the *risk* at each activity gives the a priori likelihood that an object interacting at an event will eventually have a negative outcome.

In describing the details of our approach in the following chapters, we will clarify some technical background concerning rule mining, and then rigorously formalize the method. Finally, we apply the approach to a real-life log and discuss the results in light of the research goals. We start by reviewing related work.

2 Related Work

In the simplest case, an RCA in process mining can be considered as a standard ML task, using a data set where each instance has its outcome encoded in a target attribute. Existing works use general classification methods [9,10]. As in this work, rule mining has been applied for RCA [2,11,13]. Of course, instead of investigating negative outcomes, one can also foster positive process outcomes [12], in general, mine for deviances [13]. [14] proposes a general framework for analyzing process properties beyond RCA.

An important step to refine basic classification is to distinguish between correlation and causation. [3] uses structural equation models for estimating causality and assessing the impact of potential improvement actions. [1] follows a probabilistic approach with the intent to increase the robustness of findings towards spurious correlations. [12] distinguishes between controllable and non-controllable descriptive attributes in order to propose actionable treatments.

Closely related to our work are [2] and [13]. [2] clusters traces with regards to problematic attributes and extracts association rules from these clusters to discover problematic subgroups. [13] describes an approach to explain process deviances based on declarative rule and sequence mining, also taking into account the data perspective. As opposed to [2] and [13], our work focuses solely on the event-type level, but extends the scope to object-centricity. To the best of our knowledge, our work is the first to propose a pattern mining framework, as well as more specifically an RCA method on object-centric event logs.

3 Approach

In the following, we describe how we conduct this RCA. While our approach works on an object-centric log standard [6], we impose some assumptions on the structure of the input that we achieve through (semi-)automatic preprocessing. Thus, after listing preliminaries (Sect. 3.1), we give our custom specification of the input data (Sect. 3.2), before describing how this input is converted into a suitable format for rule mining (Sect. 3.3).

3.1 Preliminaries

Let X be a set. The powerset of X is denoted with $\mathcal{P}(X)$. With $\mathcal{B}(X)$, we denote the set of multisets over X. For example, $[a^4, b^1] \in \mathcal{B}(X)$ is a multiset over a set $X = \{a, b\}$ in which a occurs four times and b once. Given sets X, Y and a partial function $f : X \nrightarrow Y$, $dom(f) \subseteq X$ denotes the domain of f. *Bool* is the set of boolean values *true* and *false*.

We deploy concepts from association rule mining [8] as follows. Let again X be a set, in this context called a set of *patterns*. A *dataset* over X is $D = [T_1^{n_1}, ..., T_k^{n_k}] \in \mathcal{B}(\mathcal{P}(X))$, $k \geq 0$, where for each $i, 1 \leq i \leq k$, $T_i \subseteq X$ and $n_i \geq 1$. We call T_i the *transactions* in D. Let $X' \subseteq X$. The *support* of X' in D is defined as $supp_D(X') = \Sigma_{i=1,...,k,X' \subseteq T_i} n_i / \Sigma_{i=1,...,k} n_i$. Thus, the support of a pattern set is the fraction of transactions that include a pattern set. For $X_1, X_2 \subseteq X$, we call $r = X_1 \rightarrow X_2$ an *association rule* over X, with X_1 being the *antecedent* and X_2 the *consequent* of r. The *confidence* of r in D is defined as $conf_D(r) = supp_D(X_1 \cup X_2)/supp_D(X_1)$. The confidence of a rule gives the relative likelihood to observe the rule consequent given the antecedent. The *lift* of r is defined as $lift_D(r) = supp_D(X_1 \cup X_2)/\big(supp_D(X_1) \cdot supp_D(X_2)\big)$. The lift of a rule is a measure for the positive correlation between antecedent and consequent; in other words, it quantifies how much more likely the consequent is to appear in the context of the antecedent, compared to an a priori observation.

3.2 Input

Single source of truth for our approach is an object-centric event log. As remarked previously, the basic difference between object-centric and traditional event logs is that in the former, events may relate to an arbitrary amount of objects instead of a fixed single case. To describe these object-centric logs, we assume the following universes to be given: $\mathbb{U}_{ev}$ are events, $\mathbb{U}_{etype}$ are event types, $\mathbb{U}_{obj}$ are objects, $\mathbb{U}_{otype}$ are object types, and $\mathbb{U}_{time}$ are timestamps. For each object, a given function $otype \in \mathbb{U}_{obj} \rightarrow \mathbb{U}_{otype}$ fixes an object type. Furthermore, $\mathbb{U}_{oattr}$ are object attributes. Object attributes are assumed to resemble properties of exactly one object type, again fixed by a type signature $oatype \in \mathbb{U}_{oattr} \rightarrow \mathbb{U}_{otype}$.

Log standards such as XES or OCEL capture the data perspective of a process through event or case attributes using standard data types such as strings, booleans, and numbers. In our work, aiming for an RCA, we analyze the interplay of objects and attributes via pattern mining. Therefore, we enforce a discretization of the process data, that is, properties have to be encoded as logical propositions. On the one hand, this may impose limitations to our approach, since the question how to convert data types into a propositional form is not trivial: for example, a naive handling of continuous attributes by converting each possible value assignment to a distinct pattern is neither feasible nor sensible for association rule mining. On the other hand, attributes with a propositional encoding offer a natural way to establish expressivity for describing the interplay of objects, namely by an embedding into a first-order logic framework. That is, we regard these encodings as *predicates* in the context of events. Event attributes are regarded as predicates of arity 0, since they take no input parameters (assuming the event gives the context and is thus not a parameter itself). Object attributes are predicates of arity 1, and relations between objects are predicates of arity 2. These predicates may be assembled to formulas to be evaluated over events: Firstly, event contexts restrict the domain of predicates to the set of objects occurring at the events. Secondly, event contexts provides an intuitive interpretation, assigning truth values by looking up event attributes, and object properties at the time of event occurrence.

We would like to choose this conceptualization as the backbone of our method, since such formulas also naturally translate to patterns in the context of rule mining. That is, a formula satisfied at an event could be considered as a pattern being part of the transaction defined by the event. In order to simplify matters, however, we restrict ourselves here to consider only object attributes, that is, predicates of arity 1, using from now on the terms *predicate* and *object attribute* interchangeably. Hence, we neglect object interrelationships and event attributes, but argue that an extension is straightforward. The following is an accordingly customized and restricted definition of object-centric event data.

Definition 1 (Object-Centric Event Log). An object-centric event log (in short, OCEL) is a tuple $L = (E, O, E2O, ET, evtype, time, oattr, F, \varGamma)$ where $E \subseteq \mathbb{U}_{ev}$, $E \neq \{\emptyset\}$, are events, $O \subseteq \mathbb{U}_{obj}$ are objects, $E2O \subseteq E \times O$ are event-object relations, $ET \subseteq \mathbb{U}_{etype}$ are event types, $evtype \in E \rightarrow ET$ are event types per event, $time \in E \rightarrow \mathbb{U}_{time}$ are event timestamps, $oattr \subseteq \mathbb{U}_{oattr}$ are object attributes (predicates), and furthermore

- $F \in ET \rightarrow \mathcal{P}(oattr)$ are predicates for each event type, and
- $\varGamma \in E \rightarrow \big(\mathcal{P}(oattr) \nrightarrow (O \nrightarrow Bool)\big)$ such that for all $e \in E, et \in ET$, if $evtype(e) = et$, then $dom(\varGamma(e)) = F(et)$, and for all $oa \in dom(\varGamma(e))$, $dom(\varGamma(e)(oa)) = \{o \in O \mid otype(o) = oatype(oa) \wedge (e, o) \in E2O\}$, are predicate interpretations.

To sum up, an OCEL is a set of variously typed events (*E, evtype*) that are ordered with respect to time (*time*). Objects (O) can occur at these events (*E2O*), and the objects can carry data (*oattr*). In our approach, we treat event types as first-class citizens: at those, we investigate the interplay of objects with respect to root causes. Some attributes may hence be more or less interesting to be considered at specific event types, for example, the elapsed time of an object is uninteresting at an object creation event. Because of this, and also to restrict the set of candidates for a more efficient rule mining upfront, only a subset of the predicates describes each event type (F). Finally, an event is characterized by the selected attributes of objects that occur at these events ($\varGamma$).

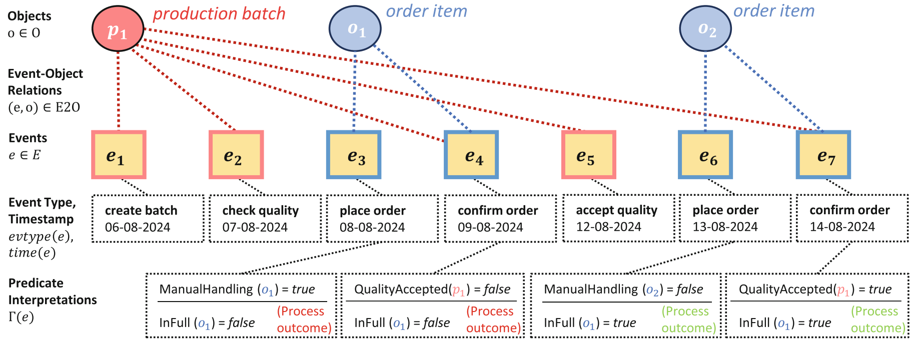

Fig. 2. An exemplary OCEL L_1.

Example 1. Figure 2 depicts an exemplary OCEL L_1. This log represents a process involving the object types *order items* and *production batches*. Here, we have:

- $E = \{e_1, ..., e_7\}$, $O = \{o_1, o_2, p_1\}$ with $otype(o_1) = otype(o_2) = order\ item$, $otype(p_1) = production\ batch$. $E2O = \{(e_1, p_1), (e_2, p_1), ...\}$, as indicated. $ET = \{create\ batch,\ check\ quality,\ accept\ quality,\ place\ order,\ confirm\ order\}$.
- Types and timestamps per event are, for example, $evtype(e_1) = create\ batch$ and $time(e_1) = 06\text{-}08\text{-}2024$, and for other events as indicated.
- $oattr = \{ManualHandling,\ QualityAccepted,\ InFull\} \subseteq \mathbb{U}_{oa}$, with $oatype(InFull) = oatype(ManualHandling) = order\ item$, and $oatype(QualityAccepted) = production\ batch$.
- $F(place\ order) = \{ManualHandling,\ InFull\}$,
- $F(confirm\ order) = \{QualityAccepted,\ InFull\}$, and
- $F(et) = \emptyset$ for $et \in \{create\ batch,\ check\ quality,\ accept\ quality\}$.
- Interpretations of predicates are, for example at e_3, $\Gamma(e_3) = \{(InFull, \{(o_1, false)\}), (ManualHandling, \{(o_1, true)\})\}$, and for other events as indicated.

The business scenario of L_1 is the same as the one for the example described in Sect. 1, depicted in Fig. 1 for a bigger underlying log. For an fexplanation on the business logic, we hence refer to the first section. Note that in L_1, *InFull* reflects an object attribute that we associate with the process outcome. For our RCA method, we impose two important assumptions on input logs. First, the categorization of process outcomes with regards to the business problem has to be encoded in such an attribute on the object level. Second, this outcome has to be known for each object and thus recorded (retrospectively) in the log.

In the following, we describe our approach to extract rules from such event logs that are useful for an RCA, using above log as a running example.

3.3 Pattern Mining

The artificial example L_1 reflects two explanations for negative outcomes in the process it describes. Firstly, orders that are assigned for manual instead of automatic handling are associated with more processing mistakes, e.g., picking wrong quantities. Secondly, orders assigned to unfinished production batches may face the issue that the supposed batch release date is not met. To unveil such correlations, we rely on mining association rules that encode the explanations (ideally, root causes reflecting real causality) in the rule antecedent and the process outcome in the consequent. To this end, input event data has to be transformed into a database for rule mining. For this, the concrete predicate interpretations observed in the log have to be abstracted into generic patterns. As mentioned, we proceed by constructing formulas over the predicates by means of common logical connectives. In the log context, these formulas are then interpreted at events with regards to the predicate interpretation qualifying the event.

In the following, *well-formed first-order logic formulas* over predicates can be constructed by means of the binary conjunction and disjunction operators $\wedge, \vee$, the unary negation operator $\neg$, and existential and universal quantifiers $\exists, \forall$.

Definition 2 (Pattern). Let L be an object-centric event log with events E, event types ET and predicates F. A *pattern* over L is a function $p \in E \nrightarrow Bool$. With P_L, we denote the set of all patterns over L. For each $et \in ET$, the *pattern formulas* $P_{L,et} \subseteq P_L$ are the well-formed first-order logic formulas over $F(et)$.

For example, $\exists\, o : InFull(o)$ and $\exists\, o : \neg\big(InFull(o)\big)$[1] are pattern formulas at the event type *place order* in L_1. In the context of e_3, for instance, $\exists\, o : \neg\big(InFull(o)\big)(e_3) = true$ because there exists an order o at the event, namely o_1, where the *InFull* predicate does not evaluate to true.

Since many different formulas, and usually also many non-equivalent formulas can be constructed over a set of predicates, we also need to make a *selection* of potentially interesting candidate patterns for conducting the RCA.

Definition 3 (Pattern Selection). Let L be an object-centric event log with event types ET and patterns $P_{L,et}$ for each $et \in ET$. A *pattern selection* is a function $psel \in ET \rightarrow \mathcal{P}(P_L)$, such that for all $et \in ET$, $psel(et) \subseteq P_{L,et}$.

Given this, we can transform an OCEL into datasets for rule mining.

Definition 4 (OCEL to Pattern Datasets). Let L be an object-centric event log with events E, event types ET and patterns P_L. Furthermore, let $psel \in ET \rightarrow \mathcal{P}(P_L)$ be a pattern selection. The pattern datasets $\mathcal{D}_{L,psel} \in ET \rightarrow \mathcal{B}\big(\mathcal{P}(P_L)\big)$ corresponding to the event types in L are defined as $\mathcal{D}_{L,psel}(et) = [\{p \in psel(et) \mid p(e)\} \mid e \in E, evtype(e) = et]$.

Example 2. For L_1, Table 1 lists the pattern datasets $\mathcal{D}_{L_1,psel}(et)$ for $et \in \{$*place order, confirm order*$\}$. Here, *psel* selects the indicated formulas (that is, patterns) over the predicates *ManualHandling*, *InFull* at *place order*, and at *confirm order* over *QualityAccepted*, *InFull*. Consider for example event e_3 (*place order*). Here, the two patterns $\exists\, o : ManualHandling(o), \exists\, o : \big(\neg\, InFull(o)\big)$ are satisfied, yielding the corresponding transaction in the pattern dataset. Since no other *place order* event satisfies the very same patterns, the corresponding count is 1, as for all other listed transactions (indicated by set superscripts).

[1] We assume that variables are implicitly typed through the attribute typing *oatype*. For example, o is of type *sales order* because this is the *oatype* of *InFull*.

Table 1. Pattern datasets for activities in the exemplary log L_1.

Event Type *et*	Dataset $\mathcal{D}_{L_1,psel}(et)$
place order	$[\{\exists\, o: \mathit{ManualHandling}(o), \exists\, o: \big(\neg\, \mathit{InFull}(o)\big)\}^{1}$, $\{\exists\, o: \neg\, \big(\mathit{ManualHandling}(o)\big), \exists\, o: \mathit{InFull}(o)\}^{1}]$
confirm order	$[\{\exists\, p: \neg\big(\mathit{QualityAccepted}(p)\big), \exists\, o: \big(\neg\, \mathit{InFull}(o)\big)\}^{1}$, $\{\exists\, p: \mathit{QualityAccepted}(p), \exists\, o: \mathit{InFull}(o)\}^{1}]$

Consider now $D_1 = \mathcal{D}_{L,epat}(\mathit{place\ order})$ and $D_2 = \mathcal{D}_{L,epat}(\mathit{confirm\ order})$. Let $r_1 = \emptyset \rightarrow \{\exists\, o: \big(\neg\, \mathit{InFull}(o)\big)\}$ be an association rule. In both datasets, we have the rule confidence (risk) $\mathit{conf}_{D_1}(r_1) = \mathit{conf}_{D_2}(r_1) = 0.5$. Speaking in terms of an RCA in the domain, this is the a priori probability that an order will eventually be rejected at the time when *place order* and *confirm order* happen, respectively. Consider now the rule $r_2 = \{\exists\, p: \neg\, \mathit{QualityAccepted}(p)\} \rightarrow \{\exists\, o: \big(\neg\, \mathit{InFull}(o)\big)\}$. We have $\mathit{lift}_{D_2}(r_2) = 2$. This indicates that the risk value doubles (in absolute terms) if at the time when *confirm order* is executed, the quality of the production batch corresponding to the order has not yet been accepted.

Hence, the rationale of our approach is to analyze root causes of negative process outcomes *(a)* along the process, that is, for each activity of interest and *(b)* across all relevant object types of interest. For this, we deploy association rules where the problematic outcome is encoded in the rule consequent (in the following called *target pattern*), and the rule antecedents (in the following called *descriptive patterns*) reflect possible explanations. In the next chapter, we describe the implementation of our solution.

4 Implementation

We implemented facilities for the approach presented here, also provided via Github[2]. In the following, we describe the application pipeline of the tool **(1–2)**, describe how the results can be put into the context of a process model **(3)**, and give heuristics to select interesting rules from the mined rules **(4)**.

1) Log Preprocessing. Our tool accepts a log in the OCEL2.0 [6] standard. Upon upload, predicates are derived from the observations. As mentioned earlier, we are not restricted to capture only object attributes, but may also encode event attribute assignments and object relationships. Attributes are only regarded if the number of unique labels (values) does not exceed a user-defined threshold. This applies to attributes of any type, both nominal and non-nominal.

2) Pattern Search. By default, existentially quantified formulas are constructed over object attributes, comparable to the examples shown in the previous section. Also, formulas describing event attributes are constructed. Our tool

[2] https://github.com/beneknopp/interaction-pattern-mining.

allows to specify custom formulas over the available predicate symbols. Besides a selection of descriptive patterns, the target pattern has to be specified. After selecting patterns for each event type of interest, association rules are mined [8], parametrized by a minimal support of individual patterns.

3) Presentation. In order to comprehensively report the mining results, we suggest to consider a frequency-annotated *object-centric directly-follows graph (OCDFG)* [7], as depicted in Fig. 1. An OCDFG is the generalization of a traditional DFG involving multiple object types. Each activity shows the frequency of occurrences, i.e., the number of events of that activity. Each typed edge between two activities is also annotated with the number of times this directly-follows relation was realized by an object of the respective type. Here, we report at each event type as *risk* the confidence of the rule $\emptyset \rightarrow \{p\}$, where p is the target pattern. That is the a priori likelihood that at event time, the process will eventually have a negative outcome. The annotated model may be helpful in putting the rules at event types into the context of the whole process.

5 Experiments

To show the usefulness our approach, we investigated the order-to-cash process of a globally acting food and drink processing company. The used (confidential) object-centric log describes a time span of 1.5 years and is based on SAP data from multiple dimensions. We focus here on the object types *sales order items, delivery items, delivery headers* and *production batches.* A sales order item is (transitively) linked to objects of other types. Namely, each item has one to many delivery items, each delivery item has exactly one header and draws supply from exactly one production batch. However, by filtering, we select only sales order items that are linked to exactly one delivery item. We choose this filter in order to facilitate interpreting the mined rules, since each existentially quantified formula in a rule, for each event, binds to an unambiguous reference object. Thus, measurements are not obfuscated by varying object type multiplicities.

As the use case, we investigated the *On-Time-In-Full (OTIF)* KPI of the order-to-cash process. OTIF measures the degree to which orders were delivered timely and completely to customers. On the level of object types, this KPI can by definition be decomposed into the *OnTime* property on delivery items and the *InFull* property on sales order items. We considered a binary representation of these subproperties, comparable to the *InFull* example presented in the previous sections, so each sales order item is either *InFull* or not and each delivery item is either *OnTime* or not. Findings for the *OnTime* property are shown in Table 2 and for the *InFull* property in Table 3.

The exemplary findings for *OnTime* in Table 2 are based on a sample of 10.000 delivery items and apply to instances at the time of the *perform batch split* activity, at which point production supply is assigned to the item. This activity usually happens shortly before the actual delivery. The first depicted rule shows that 25% of all batch splits are performed on a *Friday*, and in this case, the risk of an untimely delivery was 1.89 times higher. The other two depicted rules

Table 2. Rules for the *OnTime* property at the activity *perform batch split*, on a data set of 10.000 records (events). The risk of the target pattern $\exists\, i : \neg\, OnTime(i)$ is 0.64%. Variable i denotes the delivery item, o the sales order item and d the delivery header. The rules narrow down to a specific subset of delivery items that had a negative outcome because of inaccurate inventory slot availability forecasts for weekends.

Pattern	Support	Lift
$\{Friday()\}$ $\rightarrow \{\exists\, i : \neg\, OnTime(i)\}$	0.25	1.89
$\{Friday(),$ $\exists\, o : DaysSinceOrderCreation = 3(o)\}$ $\rightarrow \{\exists\, i : \neg\, OnTime(i)\}$	0.15	27.65
$\{Friday(),$ $\exists\, o : DaysSinceOrderCreation = 3(o),$ $\exists\, d : DaysUntilDeliveryDue = 0(s)\}$ $\rightarrow \{\exists\, i : \neg\, OnTime(i)\}$	0.01	39.91

Table 3. Rules for the *InFull* property at the activity *confirm order* across a segmentation of the process with regards to material types. Variable o denotes the sales order item and p the associated production batch. Each cell in a pattern row shows the support on top and the lift below. The rules show that root causes with respect to the production dimension are prevalent and vary across the different material types.

Material	1	2	3	4	5	6	7
Event (Object) Count	4.5k	1.5k	2.0k	800	700	1.7k	2.0k
Risk	1.7%	1.3%	2.7%	5.4%	7.8%	1.2%	2.2%
$\{\exists\, p : IssueOnTime(p)\}$ $\rightarrow \{\exists\, o : \neg\, InFull(o)\}$	0.01 0.00	0.12 1.57	0.01 4.06	0.06 0.77	0.08 0.87	0.00 0.00	0.05 2.45
$\{\exists\, p : Underproduction(p)\}$ $\rightarrow \{\exists\, o : \neg\, InFull(o)\}$	0.52 1.30	0.64 1.17	0.46 1.35	0.34 1.15	0.33 1.23	0.56 1.37	0.67 0.93
$\{\exists\, p : MotherBatchScrap(p)\}$ $\rightarrow \{\exists\, o : \neg\, InFull(o)\}$	0.10 0.37	0.04 6.50	0.08 1.80	0.01 3.1	0.01 0.00	0.05 1.56	0.05 1.42
$\{\exists\, p : PlantX(p)\}$ $\rightarrow \{\exists\, o : \neg\, InFull(o)\}$	0.32 0.12	0.0 0.0	0.61 1.06	0.01 0.00	0.01 0.0	0.05 1.91	0.05 2.00

show that this risk strongly increases in that context if also the associated order had been placed several days before, and furthermore, the delivery is due on the day of event time. Discussing the details of the affected process instances with domain experts, we found out that the negative outcomes were due to not enough inventory slots being available at the customers' side, which in turn was due to inaccurate forecasts of inventory availabilities several days in advance. In consequence, delivery had to be shifted past the originially requested date. While this true root cause is not - and, in fact, cannot be - represented by means of the object-centric patterns since the data is scoped intra-organizationally, the patterns provide a minimal discriminating description of the affected instances.

The rules in Table 3 are based on a segmentation of the process across various material types *1,...,7*, and associate the *InFull* outcome of sales order items of these material types with various patterns on related production batches at time of *confirm order*. Again, there is a one-to-one correspondence between observations (events) and objects (sales order items). The descriptive patterns comprise: indicators that the batch had an on-time issue in production (*IssueOnTime*), that less was produced than planned (*Underproduction*), that there is a related mother batch that was scrapped (*MotherBatchScrap*), and that the batch was produced at a specific plant (*PlantX*). As compared to the *OnTime* rules in Table 2, the findings for *InFull* do not pinpoint to specific instances; rather, they give a broad overview on how different factors in production upstream might have an influence on the order fulfillment. Here, we remark that threads to validity partly exist due to rather low risk and support values of some of the rules.

6 Discussion

The experimental results show the strength of the approach and, in particular, the added value of the surrounding object-centric framework, in conducting an RCA. This is exemplified by all rules depicted in Tables 2 and 3, since they relate the outcome captured on the level of one object type to descriptive patterns at other object types. In particular, the rules of greater length in Tables 2, by linking together multiple patterns, show the usefulness of mining for sets of patterns for the RCA. Furthermore, note that information across several object types (sales order items, delivery headers and items) was required here to find a description of minimal length for the set of problematic instances. In summary, we argue that we have found evidence that bad process outcomes can in fact be explained via interacting objects, as aimed for in our research goal.

A limitation of the approach is that rules can be hard to interpret if formulas refer to object types of varying cardinalities. This challenge is inherent to the object-centric framework that allows for this variability. As stated, we dispelled such ambiguities in our experiments by pre-filtering the input. Another shortcoming of our approach follows from the inherent limitation of itemset mining to discretized attributes. Here, we see potential for refinement in future work, for example by adapting the mechanisms of the C4.5 algorithm well-known from decision tree learning for discretizing numerical and continuous attributes.

7 Conclusion

In this paper, we have introduced a novel method for root cause analyses (RCA) on object-centric logs using association rule mining. The experiments have substantiated the usefulness of that method. In our opinion, the results suggest a synergy between pattern mining and object-centric process mining.

The first-order logic based query engine implemented here is not restricted in its use to RCA. We argue that other use cases are possible, as long as these can be formulated on the event level. For example, one could validate constraints at event types using arbitrary complex formulas. In future work, we would like to explore via case studies whether complex root causes can be found in real processes that further leverage the presented logical query engine. Also, we would like to improve the extent to which the object-centric rules can provide operational support by embedding them into a causal inference framework [3,12]. Finally, we would like to explore whether pattern-based descriptions can be used for creating more realistic discrete event simulation models [15].

References

1. Van Houdt, G., Depaire, B., Martin, N.: Root cause analysis in process mining with probabilistic temporal logic. In: Munoz-Gama, J., Lu, X. (eds.) ICPM 2021. LNBIP, vol. 433, pp. 73–84. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-98581-3_6
2. Fani Sani, M., van der Aalst, W., Bolt, A., García-Algarra, J.: Subgroup discovery in process mining. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 237–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_17
3. Qafari, M.S., van der Aalst, W.: Root cause analysis in process mining using structural equation models. In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) BPM 2020. LNBIP, vol. 397, pp. 155–167. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66498-5_12
4. van der Aalst, W.: Object-centric process mining: unraveling the fabric of real processes. Mathematics **11**(12), 2691 (2023)
5. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: Process Mining Handbook. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_9
6. Koren, I., Adams, J.N., Berti, A.: OCEL 2.0 Resources–www. ocel-standard. org. arXiv preprint arXiv:2403.01982 (2024)
7. Berti, A., van der Aalst, W.: Extracting multiple viewpoint models from relational databases. In: Ceravolo, P., van Keulen, M., Gómez-López, M.T. (eds.) SIMPDA 2018-2019. LNBIP, vol. 379, pp. 24–51. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46633-6_2
8. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (1993)
9. Gupta, N., Anand, K., Sureka, A.: Pariket: mining business process logs for root cause analysis of anomalous incidents. In: Chu, W., Kikuchi, S., Bhalla, S. (eds.) DNIS 2015. LNCS, vol. 8999, pp. 244–263. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16313-0_19
10. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 174–186. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_18
11. Böhmer, K., Rinderle-Ma, S.: Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users. Inf. Syst. **90**, 101438 (2020)
12. Bozorgi, Z.D., Teinemaa, I., Dumas, M., La Rosa, M., Polyvyanyy, A.: Process mining meets causal machine learning: discovering causal rules from event logs. In: 2nd International Conference on Process Mining (ICPM). IEEE (2020)

13. Bergami, G., di Francescomarino, C., Ghidini, C., Maggi, F.M., Puura, J.: Exploring business process deviance with sequential and declarative patterns. arXiv preprint arXiv:2111.12454 (2021)
14. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general framework for correlating business process characteristics. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 250–266. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_16
15. Knopp, B., Pourbafrani, M., van der Aalst, W.: Discovering object-centric process simulation models. In: 5th International Conference on Process Mining (ICPM). IEEE (2023)