

Challenges of Anomaly Detection in the Object-Centric Setting: Dimensions and the Role of Domain Knowledge

Alessandro Berti^{1,2}, Urszula Jessen^{3,4}, Wil M.P. van der Aalst^{1,2}, Dirk Fahland⁴

¹ Process and Data Science Chair, RWTH Aachen University, Aachen, Germany

² Fraunhofer FIT, Sankt Augustin, Germany

³ Process Insights, ECE Group Services, Hamburg, Germany

⁴ Eindhoven University of Technology, The Netherlands

{a.berti, wvdaalst}@pads.rwth-aachen.de; {u.a.jessen, d.fahland}@tue.nl

Abstract. Object-centric event logs, allowing events related to different objects of different object types, represent naturally the execution of business processes, such as ERP (O2C and P2P) and CRM. However, modeling such complex information requires novel process mining techniques and might result in complex sets of constraints. Object-centric anomaly detection exploits both the lifecycle and the interactions between the different objects. Therefore, anomalous patterns are proposed to the user without requiring the definition of object-centric process models. This paper proposes different methodologies for object-centric anomaly detection and discusses the role of domain knowledge for these methodologies. We discuss the advantages and limitations of Large Language Models (LLMs) in the provision of such domain knowledge. Following our experience in a real-life P2P process, we also discuss the role of algorithms (dimensionality reduction+anomaly detection), suggest some pre-processing steps, and discuss the role of feature propagation.

Keywords: Object-Centric Anomaly Detection · Object-Centric Feature Extraction · Procurement Processes · Large Language Models

1 Introduction

Process mining, a branch of data science, derives insights from data recorded by information systems on business processes. Traditional techniques assume each event is linked to a single case, causing repeated data extractions, minimal consideration of the interactions between different object types, and issues like deficiency (events excluded if they don't fit the case notion), convergence (events related to several objects of the same object type are replicated in different cases), and divergence (events in a case having the same activity may be related to different objects, leading to misleading causalities) [1]. Object-Centric Process Mining (OCPM) eliminates the single-case assumption, accommodating events involving various object types. Several techniques for process discovery [2] and

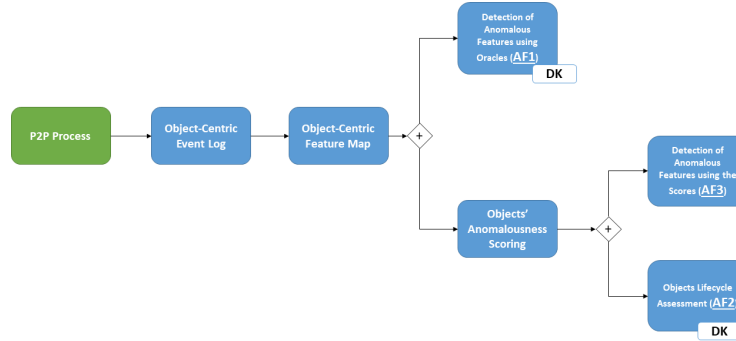


Fig. 1: Outline of the contributions proposed in the paper. The approaches highlighted with “DK” require domain knowledge.

conformance checking [13,18] have been proposed in the object-centric setting. In practice, the main challenge is the large number of object types in real-life logs. For example, a Purchase-to-Pay (P2P) log includes multiple object types such as purchase requisitions, orders, invoices, and payments, and their interactions. It is challenging to define process models or constraints that cover the lifecycle and interactions of numerous object types. Additionally, assessing results from mainstream conformance checking techniques is difficult [12], with techniques often focusing on just one perspective: control-flow, time or data.

Anomaly detection [19] is the process of identifying data points, events, or observations that deviate significantly from the expected pattern within a dataset. This technique is essential in various fields, including fraud detection, network security, fault detection, and medical diagnostics. *Anomaly detection in the OCPM setting* focuses on identifying business objects that behave differently from others. By analyzing interactions, anomalies such as discrepancies in quantities and amounts between invoices and orders can be detected. Additionally, integrating temporal data enables anomaly detection based on workload variations. Challenges arise when anomaly detection spans multiple features and interactions among different object types. The computational complexity increases with the addition of features, and the complexity of interpreting results manually becomes substantial.

In this paper, we discuss challenges in object-centric anomaly detection, focusing on methodologies for obtaining explainable and actionable insights. We introduce techniques that use domain knowledge to identify *contextual outliers* and methods for interpreting anomalies without domain knowledge (*structural outliers*). This includes a review of prevalent anomaly detection and feature selection methods. Our analysis used a real-life P2P process event log, employing the *pm4py* process mining library and the *OC-PM* tool. Figure 1 summarizes the paper’s contributions. We explore methods to identify anomalous features:

AF1 We introduce an *oracle* that evaluates features with their numerical values to determine anomalies. This oracle utilizes domain knowledge⁵.

⁵ It can be either a human analyst or an LLM

- AF2 An anomaly detection algorithm assigns scores to objects, allowing a domain knowledge owner to examine the lifecycles of objects with the lowest scores for detailed anomaly patterns analysis.
- AF3 The anomaly scores of objects can be aggregated to generate a feature-level anomaly score.

The paper is structured as follows: Section 2 reviews related work. Section 3 defines key concepts for object-centric feature extraction. Section 4 outlines methods for detecting anomalous object-centric features. Section 5 discusses a case study. Section 6 concludes the paper.

2 Related Work

Feature Extraction and Anomaly Detection: Fundamental work on feature extraction and machine learning for traditional event logs is discussed in [15]. In the object-centric context, graph-based feature extraction from event logs is detailed in [5], identifying various P2P process issues such as maintenance contracts and maverick buying. These features, integrated into *pm4py* [9] and *OC-PM* [4], support various machine learning algorithms. The approach described in [6] uses these features for anomaly detection among other applications. Anomaly detection methods in process mining are reviewed in [14], with [11] utilizing Large Language Models for semantic anomaly detection, and [10] focusing on identifying and explaining anomalies' root causes.

Object-Centric Conformance Checking: The approach described in [13] divides object-centric conformance checking into lifecycle-based and interaction-based analysis. In [16], object-centric behavioral constraint models are introduced as declarative rules for activity behaviors and relationships, yet lacking a conformance checking approach. In [18], rules are verified on object lifecycles and interactions, but these rules must be manually defined without a discovery method. Object-centric Petri nets for conformance checking are discussed in [2]. In particular, [17] uses object-centric alignments to match event log behavior with a Petri net model, and [3] defines fitness and precision in this context.

3 Preliminaries

In this section, we present some of the basic concepts used in the rest of the paper.

3.1 Object-Centric Event Logs

Object-centric event logs relax the assumption that an event is related to a single case notion. Instead, an event can be related to different objects of different object types.

Definition 1 (Universes). *We define the following universes: U_Σ is the universe of strings (with $<_\Sigma$ being the lexicographic order); $U_{OT} \subseteq U_\Sigma$ is the universe of object*

types; $U_O \subseteq U_\Sigma$ is the universe of objects (identifiers); $U_E \subseteq U_\Sigma$ is the universe of events (identifiers); $U_{act} \subseteq U_\Sigma$ is the universe of activities (i.e., event types); $U_{att} \subseteq U_\Sigma$ is the universe of attribute names; $U_{timest} \subseteq \mathbb{R}^+$ is the universe of timestamps; U_{val} is the universe of attribute values.

Definition 2 presents the definition of object-centric event log, requiring the universes introduced in Definition 1.

Definition 2 (Object-Centric Event Log). *An object-centric event log is a tuple $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$ in which: $E \subseteq U_E$ is the set of events; $O \subseteq U_O$ is the set of objects; $\pi_{otyp} : O \rightarrow U_{OT}$ maps each object to an object type; $\pi_{act} : E \rightarrow U_{act}$ maps each event to an activity; $\pi_{time} : E \rightarrow U_{timest}$ maps each event to a timestamp; $\pi_{omap} : E \rightarrow \mathcal{P}(O)$ maps each event to a set of related objects; $\pi_{vmap} : E \rightarrow (U_{att} \dashv U_{val})$ maps each event to an attribute map (associating a name to a value); $\pi_{ovmap} : O \rightarrow (U_{att} \dashv U_{val})$ maps each object to an attribute map; $<$ defines a total order on the events.*

The total order $<$ is based on the timestamp and the lexicographic order between the event identifiers.

Definition 3 (Auxiliary Object-Centric Definitions). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, we define:*

- **Lifecycle of an Object** $lif : O \rightarrow \mathcal{P}(E)$, $lif(o) = \{e \in E \mid o \in \pi_{omap}(e)\}$
- **Start/End events for the Lifecycle of an Object** $start(o) = \operatorname{argmin}_{<} lif(o)$, $end(o) = \operatorname{argmax}_{<} lif(o)$
- **Eventually-Follows Graph for an Object** $efg : O \rightarrow \mathcal{P}(E \times E)$, $efg(o) = \{(e_1, e_2) \in lif(o) \times lif(o) \mid e_1 < e_2\}$.
- **Directly-Follows Graph for an Object** $dfg : O \rightarrow \mathcal{P}(E \times E)$, $dfg(o) = \{(e_1, e_2) \in efg(o) \mid \nexists e_3, o \in \pi_{omap}(e_3) \ e_1 < e_3 < e_2\}$.
- **Objects of a given Object Type** For any $ot \in U_{OT}$, $O_{ot} = \{o \in O \mid \pi_{otyp}(o) = ot\}$
- **Objects Interaction** $interact : O \rightarrow \mathcal{P}(O)$, $interact(o) = \{o' \in O \mid \exists e \in E \ o \in \pi_{omap}(e) \wedge o' \in \pi_{omap}(e)\}$. For any $ot \in U_{OT}$, $interact_{ot} : O \rightarrow \mathcal{P}(O_{ot})$, $interact_{ot}(o) = \{o' \in interact(o) \mid \pi_{otyp}(o') = ot\}$.
- **Objects Creation** For any $ot \in U_{OT}$, $creation_{ot} : O \rightarrow \mathcal{P}(O_{ot})$, $creation_{ot}(o) = \{o' \in interact_{ot}(o) \mid \pi_{time}(start(o)) < \pi_{time}(start(o'))\}$.
- **Objects Continuation** For any $ot \in U_{OT}$, $continuation_{ot} : O \rightarrow \mathcal{P}(O_{ot})$, $continuation_{ot}(o) = \{o' \in interact_{ot}(o) \mid \pi_{time}(end(o)) = \pi_{time}(start(o'))\}$.
- **Objects Co-birth** For any $ot \in U_{OT}$, $cobirth_{ot} : O \rightarrow \mathcal{P}(O_{ot})$, $cobirth_{ot}(o) = \{o' \in interact_{ot}(o) \mid \pi_{time}(start(o)) = \pi_{time}(start(o'))\}$.
- **Objects Co-death** For any $ot \in U_{OT}$, $codeath_{ot} : O \rightarrow \mathcal{P}(O_{ot})$, $codeath_{ot}(o) = \{o' \in interact_{ot}(o) \mid \pi_{time}(end(o)) = \pi_{time}(end(o'))\}$.
- **Common Attributes for the Objects of a given Object Type** For any $ot \in U_{OT}$, $OATT_{ot} = \{a \in U_{att} \mid a \in \operatorname{dom}(\pi_{ovmap}(o)) \ \forall o \in O_{ot}\}$

Definition 3 outlines key concepts such as *lifecycle* and establishes the directly- and eventually-follows graph. It also forms associations through various object interactions, initially discussed in [5]. An example is the object co-birth graph, linking objects that start their lifecycle simultaneously. In the case study [6], these interactions help identify objects with incomplete lifecycles and detect orders lacking associated payments. Additionally, we define $OATT_{ot}$ as the set of attributes applicable to all objects of a specific type.

3.2 Object-Centric Feature Maps

To apply machine learning algorithms to object-centric event logs, we need to convert them to a set of numerical features. To extract such numerical features, we report the methodology introduced in [5].

Definition 4 (Object-Centric Feature Map). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, an object type $ot \in U_{OT}$, and a set of strings $\Sigma \subseteq U_{\Sigma}$, a feature map is a function $O_{ot} \rightarrow (\Sigma \rightarrow \mathbb{R})$.*

First, we introduce in Definition 4 a generic definition of object-centric feature map. Then, in Definition 5, we introduce an example object-centric feature map computed using the definitions introduced in Definition 3.

Definition 5 (Example of Object-Centric Feature Map). *Let $+$ be the string concatenation operator. Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$ and an object type $ot \in U_{OT}$, we define $F_{ot} : O_{ot} \rightarrow (\Sigma \rightarrow \mathbb{R})$ such that:*

- **Numeric Attribute Values** *For any $att \in OATT_{ot}$, if $v = \pi_{ovmap}(o)(att) \in \mathbb{R}$, then $F_{ot}(o)(\text{"numvalue"} + att) = v$*
- **One-Hot Encoding of String Attribute Values** *For any $att \in OATT_{ot}$, if $v = \pi_{ovmap}(att) \in U_{\Sigma}$, then $F_{ot}(o)(\text{"strvalue"} + att + \text{"_"} + v) = 1$*
- **Count of the Activities** *For any $a \in U_{act}$, $F_{ot}(o)(\text{"lifecyclecontains"} + a) = |\{e \in \text{lif}(o) \mid \pi_{act}(e) = a\}|$*
- **One-Hot Encoding of the Start Activities** *For any $a \in U_{act}$, $F_{ot}(o)(\text{"lifecyclestartswith"} + a) = \mathbb{1}_{a=\pi_{act}(\text{start}(o))}$*
- **Lifecycle Start Time** *$F_{ot}(o)(\text{"lifecyclestarttime"}) = \pi_{time}(\text{start}(o))$*
- **Lifecycle End Time** *$F_{ot}(o)(\text{"lifecycleendtime"}) = \pi_{time}(\text{end}(o))$*
- **Lifecycle Duration** *$F_{ot}(o)(\text{"lifecycleduration"}) = \pi_{time}(\text{end}(o)) - \pi_{time}(\text{start}(o))$*
- **Directly Follows Graph** *For any $a_1, a_2 \in U_{act}$, $F_{ot}(o)(\text{"dfg-"} + a_1 + \text{"_"} + a_2) = |\{(e_1, e_2) \in \text{dfg}(o) \mid \pi_{act}(e_1) = a_1 \wedge \pi_{act}(e_2) = a_2\}|$*
- **Number of Interactions for a given Object Type** *For any $ot' \in U_{OT}$, $F_{ot}(o)(\text{"interactions"} + ot') = |\text{interact}_{ot'}(o)|$*
- **Number of Creations for a given Object Type** *For any $ot' \in U_{OT}$, $F_{ot}(o)(\text{"creation"} + ot') = |\text{creation}_{ot'}(o)|$*

We assume Σ to contain at least the aforementioned features.

In Definition 5, we distinguish between features related to the *object attributes*, features related to the *lifecycle of an object*, and features related to the *interactions between the objects*. However, the features of neighboring objects are still not exploited. An approach to resolve such limitation is proposed in Definition 6.

Definition 6 (Feature Propagation). *Let $+$ be the string concatenation operator. Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$ two object types $ot, ot' \in U_{OT}$, and two feature maps $F_{ot} : O_{ot} \rightarrow (\Sigma_1 \rightarrow \mathbb{R})$, $F_{ot'} : O_{ot'} \rightarrow (\Sigma_2 \rightarrow \mathbb{R})$, we define a propagated feature map $F'_{ot,agg} : O_{ot} \rightarrow (\Sigma_3 \rightarrow \mathbb{R})$ where:*

- *$agg : \mathcal{B}(\mathbb{R}) \rightarrow \mathbb{R}$ is an aggregation function (for instance, the mean or the median).*

- $\Sigma_3 = \Sigma_1 \cup \{ \text{"prop"} + \sigma_2 \mid \sigma \in \Sigma_2 \}$
- For any $o \in O_{ot}$ and $\sigma_1 \in \Sigma_1$, $F'_{ot,agg}(o)(\sigma_1) = F_{ot}(o)(\sigma_1)$
- For any $o \in O_{ot}$ and $\sigma_2 \in \Sigma_2$, $F'_{ot,agg}(o)(\text{"prop"} + \sigma_2) = agg(\{F_{ot'}(o')(\sigma_2) \mid o' \in interact_{ot'}(o)\})$

Definition 6 allows merging two feature maps, considering both the features of an object (attributes+lifecycle+interactions) and an aggregation of the features of the neighboring objects.

4 Approach

The section detail three methods to identify anomalous feature values. Section 4.1 uses the feature map and domain knowledge for anomaly recognition (methodology **AF1**). Section 4.2 applies anomaly detection algorithms to score objects and examines those with the lowest scores (methodology **AF2**). Section 4.3 aggregates these scores at the object level to assign scores to features (methodology **AF3**).

4.1 Anomalous Features Identification through Oracles (AF1)

Starting from an object-centric feature map, we can already get useful insights about anomalous values. A domain knowledge owner, simply looking at the feature map and its values, could spot a list of anomalous patterns in the process according to the domain knowledge (methodology **AF1**). Listing 1.1 shows an example detection of patterns.

Listing 1.1: Anomalies spotted by an LLM starting from the feature table (**AF1**)

1. **High Duration of Lifecycle**: The range of lifecycle durations is broad, with the maximum value at 722,032,120 while the 75% quantile is 1,215,387, indicating potential abnormally long lifecycles.
2. **Activity Occurrence Consistencies**: "Submit Purchase Order for Approval" and "Approve Purchase Order" activities typically occur once per object but have outliers with up to 100 occurrences. Investigate objects where these activities are repeated excessively.
3. **Degree of Centrality**: There's a significant jump in degree centrality values from the 75% quantile to the maximum. High centrality objects, potential interaction hubs, warrant further investigation.
4. **Rare Activities**: Activities like "Reject Purchase Order" and "(Re)Open Purchase Order" are infrequent. Focusing on these can reveal anomalies.
5. **Object Interactions**: Objects with an unusually high number of interactions in the graph may indicate anomalies.
6. **Lifecycle Start and Completion Time**: Analyze objects with exceptionally short or long lifecycles compared to the dataset trend.
7. **Objects Starting/Ending Lifecycle Together**: Examine cases where a notably high number of objects start or end their lifecycle simultaneously with the current object.

Listing 1.2: Anomalies spotted by an LLM on the lifecycle of a single object (**AF2**).

1. **Duplicate Time Stamps**: Both the "Approve Requisition" and "Create Purchase Order" events are recorded at the exact timestamp of 2010-10-06 03:44:22, suggesting an immediate creation of the PO after requisition approval, which is atypical due to the expected minor delay between these actions. Similarly, "Submit Purchase Order for Approval" and "Approve Purchase Order" share the timestamp of 2010-10-06 03:45:41, indicating unusual instant approval.
2. **Order of Events**: "Approve Purchase Order" is followed by another "Submit Purchase Order for Approval" event at the same timestamp, suggesting redundancy and possibly a recording error or system glitch.
3. **Long Lifecycle Duration**: The lifecycle of PO_277871 extends unusually from 2010-10-06 to 2023-07-12, closed initially on 2010-10-08 and then reopened 13 years later, which deviates from standard P2P process durations.
4. **Close and Reopen of PO**: PO_277871 was closed on 2010-10-08 and reopened on 2023-07-12, a rare occurrence that may require verification with system administrators to understand if it reflects actual procedural needs or system setup anomalies.

Definition 7 (Features’ Oracle). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, an object type $ot \in U_{OT}$, and a feature map $F_{ot} : O_{ot} \rightarrow (\Sigma \rightarrow \mathbb{R})$, we define as oracle any function $ORACLE_{\Sigma} : \Sigma \rightarrow (\mathbb{R} \rightarrow \mathbb{R})$*

Definition 7 formally introduces an “oracle” function looking at the values of the feature map and associating them with a real number. In our setting, we can assume that the oracle associates negative real numbers with anomalous feature values.

4.2 Scoring the Anomalousness of an Object (AF2)

Having an object-centric feature map, we could apply any anomaly detection algorithm (such as isolation forests or local outlier factor) to assign an anomaly score to the objects. The objects having lower anomaly score are considered anomalous.

Definition 8 (Objects’ Score Function). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$ and an object type $ot \in U_{OT}$, we define as score function any function $SCORE_{ot} : O_{ot} \rightarrow \mathbb{R}$.*

In Definition 8, we formally introduce a score function associating each object with a real number. In our setting, the score function is the anomaly detection algorithm.

Definition 9 (Objects’ Score Rank Function). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, an object type $ot \in U_{OT}$, and a score function $SCORE_{ot} : O_{ot} \rightarrow \mathbb{R}$, we define as rank any injective function $RANKSCORE_{ot} : O_{ot} \rightarrow \mathbb{N}$ such that for any $o_1, o_2 \in O, o_1 \neq o_2$:*

$$RANKSCORE_{ot}(o_1) < RANKSCORE_{ot}(o_2) \iff SCORE_{ot}(o_1) < SCORE_{ot}(o_2)$$

Object ID	Isolation Forest Scores	Local Outlier Factor Scores
PO_23667	-0.200785	-40.049412
PO_23507	-0.200311	-7.200163
PO_23508	-0.200311	-7.200163
PO_23512	-0.200311	-7.200163
PO_23513	-0.200311	-7.200163
PO_23514	-0.200311	-7.200163
PO_23515	-0.200311	-7.200163
PO_23516	-0.200311	-7.200163
PO_23517	-0.200311	-7.200163
PO_277871	-0.195874	-7.622763
PO_23511	-0.189318	-7.200163
PO_3903	-0.187092	-54.929239
PO_133097	-0.175762	-8.086049
PO_23668	-0.174838	-39.503084
PO_23669	-0.174838	-39.503084
PO_23510	-0.174382	-7.217331
PO_86355	-0.172010	-3.117746
PO_85465	-0.171363	-0.125512
PO_23518	-0.170136	-7.212333
PO_23519	-0.170136	-7.212333
PO_23520	-0.170136	-7.212333
PO_23521	-0.170136	-7.212333
PO_23522	-0.170136	-7.212333
PO_84184	-0.169095	-1.549233
PO_3836	-0.168964	-213.993317
PO_3837	-0.168964	-213.982787
PO_3838	-0.168964	-213.974041
PO_3839	-0.168964	-213.967588
PO_3840	-0.168964	-213.960370
PO_3841	-0.168964	-213.953323

Table 1: Anomaly scores for some purchase orders of the considered log.

Feature (with Value)	Count	FEA.SCORE
1 Occurrence of the activity Cancel Purchase Order	300	-0.07
1 Occurrence of the activity (Re)Open Purchase Order	167	-0.12
44 other orders are terminating with the same event	45	-0.21
45 other objects are interacting with the order	45	-0.21
The activity Approve Purchase Order is not executed	131	-0.07
There are 2 activities in the lifecycle of the order	72	-0.09
29 other orders are terminating with the same event	30	-0.19
30 other objects are interacting with the order	30	-0.19
27 other orders are terminating with the same event	28	-0.19
28 other objects are interacting with the order	28	-0.19
20 other orders are terminating with the same event	21	-0.18
There is a single event in the lifecycle of the order	53	-0.04
The activity Submit Purchase Order for Approval is not executed	53	-0.04
There are 13 events in the lifecycle of the order	41	-0.05

Table 2: Features’ values correlated with anomalies (methodology **AF3**).

The score (for instance, related to the application of an anomaly detection algorithm) is used in Definition 9 to introduce a rank between the objects. In methodology **AF2**, we propose to explore the lifecycle of the most anomalous objects with the goal of understanding the anomalous patterns. Listing 1.2 shows an example detection of patterns on the lifecycle of an object.

4.3 Anomalous Features Identification aggregating Anomaly Scores

Our methodology **AF3** aims to use the anomaly scores at the object level to automatically assign an anomaly score to the values of the feature map.

Definition 10 (Normalization of a Feature Map). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, an object type $ot \in U_{OT}$, a feature map $F_{ot} : O_{ot} \rightarrow (\Sigma \rightarrow \mathbb{R})$, and a real number $\epsilon > 0$, we define for $\sigma \in \Sigma$:*

- $min_{\sigma} = \min\{F_{ot}(o)(\sigma) \mid o \in O_{ot}\}$.
- $max_{\sigma} = \max\{F_{ot}(o)(\sigma) \mid o \in O_{ot}\}$.
- $F_{ot,\epsilon}^{norm} : O_{ot} \rightarrow (\Sigma \rightarrow [-1, 1])$,

$$F_{ot,\epsilon}^{norm}(o)(\sigma) = -1 + 2 \times \frac{F_{ot}(o)(\sigma) - min_{\sigma}}{max_{\sigma} - min_{\sigma} + \epsilon}$$

Since features' values are heterogenous (for instance, the lifecycle of the objects is measured in seconds, while one-hot encoding features are valued 0 or 1), we propose in Definition 10 an approach to normalize such values between -1 and 1 .

Definition 11 (Features' Score). *Given an object-centric event log $L = (E, O, \pi_{otyp}, \pi_{act}, \pi_{time}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, <)$, an object type $ot \in U_{OT}$, a normalized feature map $F_{ot,\epsilon}^{norm} : O_{ot} \rightarrow (\Sigma \rightarrow \mathbb{R})$ and a score function $SCORE_{ot} : O_{ot} \rightarrow \mathbb{R}$, we define $FEA_SCORE : \Sigma \rightarrow \mathbb{R}$ such that for $\sigma \in \Sigma$*

$$FEA_SCORE(\sigma) = \sum_{o \in O_{ot}} \frac{SCORE_{ot}(o) \times F_{ot,\epsilon}^{norm}(o)(\sigma)}{|O_{ot}|}$$

Definition 11 implements **AF3** by assigning a score to each feature starting from the scores at the object level. Table 2 shows an example in which the objects' anomaly scores are aggregated to provide the most anomalous features' values.

5 Case Study

In this section, we discuss the application of the proposed techniques on top of a real-life P2P object-centric event log (ECE group).

5.1 Context

ECE began using the Celonis platform for process mining in 2020, integrating systems like xFlow for document acquisition and SAP ERP. Traditional process mining faced issues such as convergence and divergence [1]. Initially, ECE used the multi-event log approach in Celonis, but later transitioned to tools and libraries from the PADS group at RWTH Aachen University. Insights from this case study were detailed in [6] and shared with stakeholders through seminars.

We are interested in applying anomaly detection to discover deviations from the expected behavior (non-compliance, such as *maverick buying*, i.e. inserting formally the order only after its placement, and *post-mortem changes to purchase requisitions*) and identify behavior leading to a monetary loss in the P2P process (for example, invoice paid double, or discount rates not taken because of invoices taking long to process, or non-justified payment blocks). The aforementioned non-compliant and/or non-optimal behavior could in principle be identified with conformance checking techniques rather than anomaly detection. However, that requires pinpointing a priori all the causes of deviations. Anomaly detection can instead be applied without requiring prior knowledge of the possible deviations.

Our analysis primarily utilized the *pm4py* process mining library [9] and the *OC-PM* Javascript-based tool [4], which both support object-centric feature extraction as outlined in [5]. In previous work, we used these tools in a case study [6]. *pm4py* provides a dataframe via `pm4py.extract_ocel.features`, compatible with any Python machine learning library. *OC-PM*⁶, after feature extraction, employs the “Isolation Forests” anomaly detection algorithm.

5.2 Methodologies and Algorithms

By applying the methodologies **AF1**, **AF2**, and **AF3**, we can identify some inherent differences. The method in **AF1** bypasses anomaly detection algorithms, reducing computational costs and domain knowledge requirements, but only evaluates single features, not their combinations. **AF2** can identify anomalies across feature combinations but requires extensive domain knowledge exploration of object lifecycles, demanding more time. **AF3** operates without domain knowledge, potentially resulting in a lengthy list of anomalous features that may challenge analysts.

The methods used in the analysis of object-centric features, including feature selection, dimensionality reduction, and anomaly detection, were selected among the most popular options. For feature selection, variance was used to retain features with significant variability, suggesting their importance in distinguishing between data points. Dimensionality reduction employed *Principal Component Analysis (PCA)* and *FastMap*, both effective in reducing the number of variables. PCA transforms data into principal components, linear combinations of the original variables, while *FastMap* is a distance-preserving projection that maps data into a lower-dimensional space. Anomaly detection involved *Isolation Forests* and *Local Outlier Factor (LOF)*, chosen for their ability to identify

⁶ <https://www.ocpm.info/>

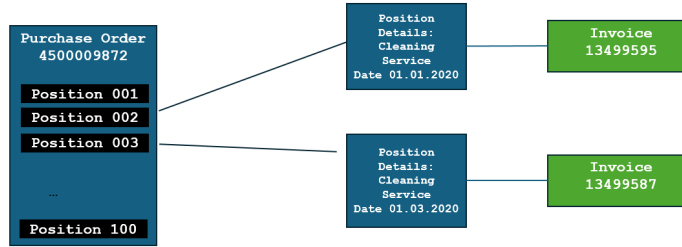


Fig. 2: Interaction between maintenance contracts with several positions and invoices.

outliers. Isolation Forests isolate anomalies by selecting a feature and a split value randomly, while LOF measures local deviation of data points from their neighbors to identify similar density regions.

In our P2P object-centric setting, FastMap was the preferred method due to its ability to maintain non-linear relationships and computational efficiency. Unlike PCA, which involves intensive eigen-decomposition and can be less suitable for large datasets with ambiguous component interpretations, FastMap efficiently reduces high-dimensional data into lower dimensions without requiring full distance matrix computations. This is particularly advantageous for managing graph-based features.

The findings highlight the strengths of Isolation Forests and LOF in anomaly detection. Isolation Forests are effective for high-dimensional data and large volumes, isolating anomalies using decision tree splittings without needing pairwise distance calculations. This accelerates anomaly detection in complex datasets. LOF excels at identifying anomalies in specific subgroups by calculating local density deviations, useful for clustered data. However, LOF requires more computational resources for large datasets. In our analysis, Isolation Forests successfully detect anomalies in object-centric event logs with traditional lifecycle features, while LOF is preferable for graph-based features, focusing on local context to identify anomalies in networks of object interactions.

5.3 Refinement of the Analysis

After performing an initial analysis, we performed some postprocessing of the object-centric event log and applied feature propagation to enhance the results.

We have hundreds of activities in the object-centric event log, mostly related to changing field values (change tables in SAP). Most of them are not relevant for object-centric anomaly detection and increase the dimensionality of the data with little gain. After our first application of anomaly detection, we repeated it on an object-centric event log that was filtered keeping only the relevant activities. The selection of relevant activities proved challenging on its own. Some infrequent activities, which were the first candidates for removal, identify important anomalies. We could distinguish between manual and automatic activities, with the latter being less important for anomaly detection.

We discovered that a traditional object-centric feature map based on the lifecycle and interactions of object types gives an incomplete process view. For instance, we found that invoices were often blocked for orders lacking preliminary purchase requisition approval, a pattern not visible when considering only invoices. By extending invoice data with information from related purchase orders (using Definition 6), we identified the root cause of this performance issue. Another observation, illustrated in Figure 2, showed that orders with multiple positions (e.g., maintenance contracts) might appear anomalous when viewed in isolation. However, considering each item’s direct relation to an invoice, such behavior is not anomalous.

5.4 Main Results

Anomaly detection allowed us to identify several non-compliance issues in the P2P process. We identified a non-negligible amount of orders with the *maverick buying* problem. The order is placed to the supplier skipping all the approval steps, the supplier sends an invoice to the company, and only then the purchase order is formally created in the ERP system. Moreover, we recorded several change activities done to purchase requisitions after their approval in order to match the amounts/quantities of the purchase order (*post-mortem changes to PRs*). This is a deleterious behavior as the purchase requisition was deliberately proposed to the managers with a lower amount.

Looking at the inefficiencies in the process leading to a monetary loss, we observed orders invoiced (and paid) several times, which were not maintenance contracts. Moreover, we identified invoices with an excessive number of change activities, signaling an inefficiency in the process (as this behavior is correlated with longer processing times). Considering the interaction between purchase orders, invoices, and payments, we observed that inefficiencies in the purchase orders also lead to inefficient processing of payments.

5.5 Limitations of LLMs as Domain Knowledge Providers

We used LLMs to interpret results, following methods in [8]. Specifically, `pm4py.llm.abstract_ocel_features` was used for textual abstraction in method **AF1**, and `pm4py.llm.abstract_ocel` for **AF3**. The *gpt-4-turbo* LLM model, available as of *09-04-2024* in Germany, was chosen for its large context window to generate insights.

Applying LLMs to textual abstractions from our object-centric event log produced mixed results. For methodology **AF1**, the insights helped identify anomalous patterns and filter objects for further analysis using the OC-PM tool.

However, several limitations arose. The context window of the LLM, despite improvements with the *gpt-4-turbo* model, restricted the inclusion of lifecycles with many events, limiting the application of methodology **AF2** to objects with fewer events. Inconsistencies across different sessions were noted [7], sometimes

requiring the merging of insights from different sessions as an "ensemble". Hallucinations and irrelevant outputs compared to the original prompt also occurred [7].

6 Conclusion

In this paper, we explored methodologies for object-centric anomaly detection and their implementation challenges in a real-life P2P process. We discovered that selecting appropriate algorithms is crucial, as they vary in effectiveness depending on the type of object-centric features. Identifying anomalous feature values is only part of the process; interpreting these results typically requires domain knowledge. Large Language Models (LLMs) can supplement domain knowledge, enabling analysts with limited process expertise to gain insights. However, challenges such as hallucinations and output inconsistency in LLMs must be noted.

References

1. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: SEFM 2019. vol. 11724, pp. 3–25. Springer (2019)
2. van der Aalst, W.M.P., Berti, A.: Discovering object-centric petri nets. *Fundam. Informaticae* **175**(1-4), 1–40 (2020)
3. Adams, J.N., van der Aalst, W.M.P.: Precision and fitness in object-centric process mining. In: ICPM 2021. pp. 128–135. IEEE (2021)
4. Berti, A., van der Aalst, W.M.P.: OC-PM: analyzing object-centric event logs and process models. *Int. J. Softw. Tools Technol. Transf.* **25**(1), 1–17 (2023)
5. Berti, A., Herforth, J., Qafari, M.S., van der Aalst, W.M.P.: Graph-based feature extraction on object-centric event logs. *International Journal of Data Science and Analytics* (2023)
6. Berti, A., Jessen, U., Park, G., Rafiei, M., van der Aalst, W.M.P.: Analyzing interconnected processes: using object-centric process mining to analyze procurement processes. *International Journal of Data Science and Analytics* (2023)
7. Berti, A., Kourani, H., Hafke, H., Yun-Li, C., Schuster, D.: Evaluating Large Language Models in Process Mining: Capabilities, Benchmarks, Evaluation Strategies, and Future Challenges. In: *Proceedings of the BPM-DS 2024 Working Conference (TBP)*. Springer (2024), <https://doi.org/10.48550/arXiv.2403.06749>
8. Berti, A., Schuster, D., van der Aalst, W.M.P.: Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. In: *BPM 2023 Workshops*. vol. 492, pp. 427–439. Springer (2023)
9. Berti, A., van Zelst, S.J., Schuster, D.: Pm4py: A process mining library for python. *Softw. Impacts* **17**, 100556 (2023)
10. 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)
11. Caspary, J., Rebmann, A., van der Aa, H.: Does this make sense? machine learning-based detection of semantic anomalies in business processes. In: *BPM 2023*. vol. 14159, pp. 163–179. Springer (2023)

12. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: a state-of-the-art literature review. In: S-BPM ONE 2019. pp. 4:1–4:10. ACM (2019)
13. Fahland, D., de Leoni, M., van Dongen, B.F., van der Aalst, W.M.P.: Behavioral conformance of artifact-centric process models. In: BIS 2011. vol. 87, pp. 37–49. Springer (2011)
14. Ko, J., Comuzzi, M.: A systematic review of anomaly detection for business process event logs. *Bus. Inf. Syst. Eng.* **65**(4), 441–462 (2023)
15. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inf. Syst.* **56**, 235–257 (2016)
16. Li, G., de Carvalho, R.M., van der Aalst, W.M.P.: Automatic discovery of object-centric behavioral constraint models. In: BIS 2017. vol. 288, pp. 43–58. Springer (2017)
17. Liss, L., Adams, J.N., van der Aalst, W.M.P.: Object-centric alignments. In: ER 2023. vol. 14320, pp. 201–219. Springer (2023)
18. Park, G., van der Aalst, W.M.P.: Monitoring constraints in business processes using object-centric constraint graphs. In: ICPM 2022 Workshops. vol. 468, pp. 479–492. Springer (2022)
19. Thudumu, S., Branch, P., Jin, J., Singh, J.J.: A comprehensive survey of anomaly detection techniques for high dimensional big data. *J. Big Data* **7**(1), 42 (2020)