Process Area Extraction by Multilevel Resource Detection for Object-Centric Process Mining

Lukas Liss and Wil M. P. van der Aalst

RWTH Aachen University, Aachen, Germany
{liss, wvdaalst}@pads.rwth-aachen.de

Abstract. Organizations manage a variety of processes at multiple levels, often simultaneously, creating complex interactions among diverse object types. This multi-level process structure challenges traditional object-centric process mining techniques due to overlapping resource utilization across different process areas. To address this complexity, this paper introduces a novel approach for multilevel resource detection in object-centric process mining, aimed at structuring and clarifying the interactions across processes that may operate on different levels of the organization. We propose a method to detect and categorize process areas based on an automatically detected resource hierarchy, using object types to detect process boundaries effectively. Our approach provides three main contributions: a theoretical framework for identifying process areas by resource levels, a publicly available implementation for practical application, and a qualitative and quantitative evaluation. The evaluations demonstrate the method's capability to enhance the clarity and interpretability of mined process models, ensuring that higher-level processes are not cluttered by unrelated lower-level activities. This method enables the detection of process areas and resource hierarchies that motivate future research of resource analysis methods that utilize these hierarchies.

Keywords: Object-Centric Process Mining · Resource Detection · TOTeM Model

1 Introduction

Process mining has the overall goal to generate valuable insights into processes [22]. In reality, processes occur rarely in isolation. Companies always have multiple types of processes running simultaneously. These different types of processes can, for example, include human resource (HR) processes, financial processes, and order management processes. Each of these processes involves multiple process participants (abstractly called objects) of different types. For example *Workers*, *HR* employees, *Orders* and *Items*. These object types interact within a process. For example, an *HR* employee and *Worker* interact when a *Worker* is promoted. To represent and investigate the interactions of different object types within a process, object-centric process mining was introduced [9]. There,

Table 1: Excerpt of the running example object-centric event log, showing the event-to-object relations (left) and the object-to-object relations (right).

evendId	time	activity	object type						source	target	qualifier
			Company	Factory	Worker	Order	Item	...	c1	w1	job
e1	01	founding	c1						w1	o1	responsible
e2	03	hire worker	c1		w1				o1	i1	contains
e3	12	create order	c1			o1	i1, i2		o1	i2	contains
e4	13	promote	c1		w1				i1	f1	origin
e5	14	produce item	c1	f1	w1		i1		i1	w1	creator
...									...		

a process consists of events that can include multiple objects of different types, allowing the tracking of the behavior and interactions of multiple objects from different object types. This enables a unified data storage for all the event data within a company.

The running example organization consists of multiple *Companies* that receive *Orders* for *Items* and uses *Workers* in different *Factories* and *Warehouses* to fulfill the orders. Additionally, they have *HR* employees to manage the workforce of the company. For this example, we consider three example processes: (a) the company lifecycle process, (b) an HR process, and (c) an order management process. Companies of this organization can be founded, and afterward, they can be closed, but don't need to be closed. The HR process spans the hiring of *Workers* and *HR* employees and their promotion. In the order management process, *Orders* with multiple *Items* are created, the items are produced, and then the orders are packed, and, in the end, sent. Each process involves multiple events that are tracked in the organisation's information system. Information systems, which are commonly used in the real world to track process activities across an organization, are not limited to a single type of process but track events for multiple types of processes, like those of the running example. Table 1 shows an excerpt of that data for the running example in an OCEL 2.0 format [6]. There we can see that event *e5* has the activity *produce item* and involved the *Company c1*, *Factory f1*, *Worker w1*, and *Item i1*. Also, additional explicit object-to-object relations are stored, like the information that *Worker w1* is responsible for *Order o1*.

However, this holistic event data creates a new challenge of selecting relevant parts for further analysis. A selection is necessary to enable comprehensible analysis and visualization, which is not possible for all the data for all the processes at once, especially for bigger and more complex enterprises. We propose process areas as a concept to select parts of the overall event data. A process area (as defined in Definition 9) selects some object types and some activities. Additionally, it allows for indicating additional object types that are not selected but act as resources for some activities. An example process area of the running example could select $\{Order, Item\}$ as the object types and $\{create\,order, produce\,item, package\,order, send\,order\}$ as activities. Also, it could indicate that, for example, both *Company*, *Factory*, and *Worker* act as

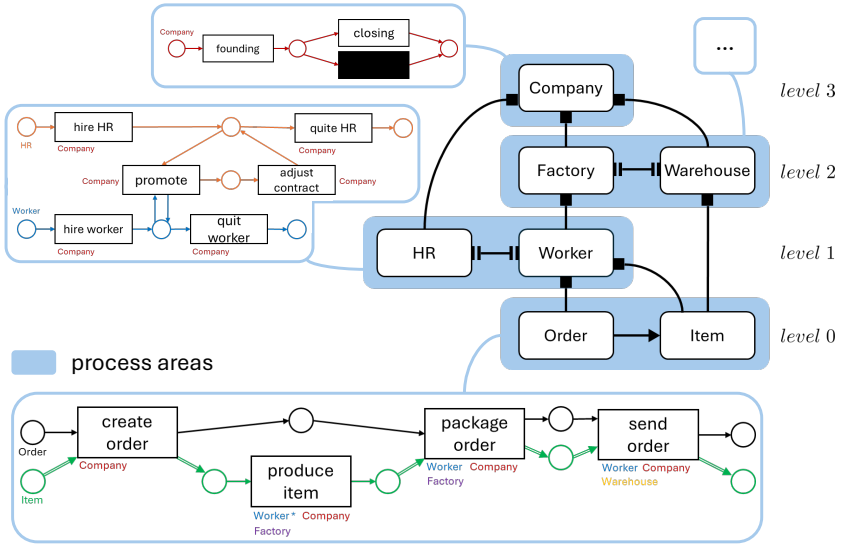

Fig. 1: Process areas for the running example. For each process area, the object-centric Petri net for relevant object types and activities is given. The activities are annotated with the resources assigned to them in that process area.

resources for the activity *produce item*. This process area, together with other process areas, of the running example are shown in Figure 1. For each process area, an object-centric Petri net is shown to visualize the behavior of that process area. In the object-centric Petri net, activities are annotated with object types that act as resources for the activities, like the *Worker* for *produce item*.

A good selection of process areas can separate complex organizational data into smaller, understandable parts, as Figure 1 shows. A key benefit of process areas is that they allow object types to act as a resource for some activities, but have their own process as well. For example, the *Worker* acts as a resource for activities involving *Orders* and *Items*, but has its own hiring process where the worker does not act as a resource but is responsible for the main workflow. Existing work in that direction focuses on the binary classification of resource objects [26], which are known to clutter process visualizations due to their highly parallel behavior. We argue that a binary classification of resources is not sufficient for enterprise-wide process analysis and advocate for a multi-level resource notion where objects can take the role of a resource for certain process areas and act as normal business objects in others. Without this separation, one would end up merging activities for which an object type acts as a business object with those where it acts as a resource. Figure 2 shows an example of such a bad selection that can happen if one mixes the activities for which the *Company* acts as a resource with those where it is a business object in the running example. The resulting process model contains a lot of undesirable parallel behavior and is therefore hard to understand.

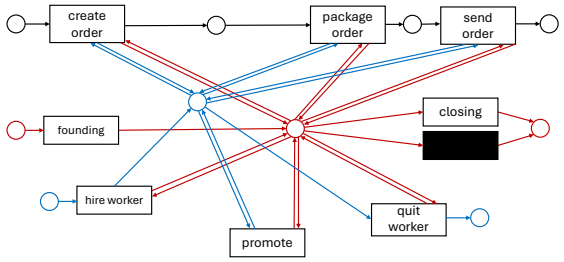

Fig. 2: Object-centric Petri net for a suboptimal subset of object types and activities showing a lot of concurrent behavior due to mixed process levels.

To automatically detect desirable process areas that lead to understandable process models without unnecessary parallelism from resource objects, like the ones in Figure 1, we propose an approach that automatically detects process areas based on multi-level resource detection. Also, we propose not to show the workflow of resources but just annotate activities that involve these resources. To do so, we group object types into levels such that higher-level object types act as resources for lower-level object types. So, for the running example, the *Company* has the highest level since it acts as a resource for all other object types. The *Worker* and *HR* employees are on a middle level, and the *Order* and *Item* are on a lower level since they are not a resource for anything else. This level assignment can also be seen in Figure 1. Our approach is based on the assumption that higher-level resources have a longer lifespan in the system than lower-level object types. For example, the company has the longest lifespan, outliving many employees, and each employee outlives many orders. We use TOTeM models [18], mined from the object-centric event log, as input to compute the resource level assignment. The temporal relation of TOTeM models describes the relation between the lifespan of objects of two types. We use this information to make sure that higher-level resources have a longer lifespan than lower-level resources.

This paper presents three contributions to enable the automatic detection of process areas for object-centric event logs, working towards enabling object-centric process mining on organization-level event logs. First, we propose an approach to detect multiple levels of resources within the object types of an organization. We use these resources to generate process areas that show behavior that occurs on the same level. Second, we provide a public implementation[1] of our approach. Third, we present a qualitative and quantitative evaluation of the proposed approach using publicly accessible object-centric event logs.

The paper is structured in the following way. In Section 2, we present related work and Section 3 gives an overview of the preliminaries. The approach for detecting process areas is presented in Section 4. Section 5 describes the qualitative and quantitative evaluation. Finally, we conclude the paper in Section 6.

[1] `https://doi.org/10.5281/zenodo.15554409`

2 Related Work

Object-centric process mining was introduced to better represent real-world processes that consist of interacting subprocesses with multiple interdependent objects [23]. Various research shows that this representation can improve the accuracy of the result in various process mining tasks, covering process discovery [24,25,18,8], conformance checking [17,13,7], monitoring [12], and prediction [10].

From the beginning on, determining relevant, coherent parts of the stored object-centric event data was a challenge [2]. For traditional process mining, this was given by the cases id, which groups together events that belong to the same case of the process [23]. In object-centric processes, no individual case notion exists that can group the events. One approach to determine which events belong together is called process executions [2]. Process executions determine groups of events in an object-centric event log based on the involved objects. One extraction technique is based on connected components of objects. It guarantees that no information loss is introduced, but may result in large process executions that cover the complete log if resource objects connect all objects of the event log [2]. The problem of resource-like objects was also addressed by van Detten et al. by detecting objects as resource-like objects and excluding them from the group-forming phase to derive advanced process executions [26]. However, it only allows for binary classification of resource-like objects on the object level and can not differentiate that some objects can act as resources for certain process areas while acting as normal objects for others. Another approach to finding groups of behavior in object-centric logs is the detection of object-centric local process models [20]. Here, many object-centric process models are discovered for subsets of activities to make models easier to understand. But the relations between potentially overlapping models on different levels of the organization remain unclear. In case-centric process mining, approaches like higher-level event abstraction and process composition try to identify parts in the process that can be grouped together [5,27]. However, all of these approaches, including the object-centric ones, assume that the event log contains one coherent process instead of the complete process data of an organization. Other related research fields, such as system composition [14] and organizational management [19], also address the issue of highly complex organizational processes by composing multiple smaller, understandable components to model the overall organization. In software architecture, the idea of microservices also supports this composition approach [4]. This composition concept is also well studied in flexible information system design, which can adapt to processes being added or removed to an organization [21]. Thus, their general approach is similar to our approach, understanding complex organizational processes as the composition of multiple smaller parts. But these approaches focus on modeling the system behavior. Identifying process areas automatically from an organization's object-centric event log remains an open research question that we address in this paper.

3 Preliminaries

An object-centric event log contains events from the universe of events $\mathbb{U}_{event}$ that have an activity from the universe of activities $\mathbb{U}_{act}$. The time of an event is from the universe of timestamps $\mathbb{U}_{time}$. Each event can be connected to multiple objects from the universe of objects $\mathbb{U}_{obj}$. The objects have a type from the universe of object types $\mathbb{U}_{type}$. Also, objects can be related to other objects via object-to-object relations. We use a subset of the OCEL 2.0 format for object-centric event logs [6].

Definition 1 (Object-Centric Event Log [6]). *$L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ is an event log with:*

- *$E \subseteq \mathbb{U}_{event}$ is a set of events, $O \subseteq \mathbb{U}_{obj}$ is a set of objects,*
- *$OT = \{\pi_{type}(o) \mid o \in O\}$ is a set of object types,*
- *$O2O \subseteq \{(o_1, o_2) \mid o_1, o_2 \in O\}$ is the set of object to object relations,*
- *$\pi_{act} : E \to \mathbb{U}_{act}$ maps each event to an activity,*
- *$\pi_{obj} : E \to \mathcal{P}(\mathbb{U}_{obj}) \setminus \{\emptyset\}$ maps each event to at least one object,*
- *$\pi_{time} : E \to \mathbb{U}_{time}$ maps each event to a timestamp.*

In an object-centric event log, objects can be connected via shared event-to-object relations and directly via object-to-object relations. To simply following notations, we use $O2O'$ for objects that are either connected via object-to-object relations or event-to-object relations.

Definition 2 (Unified Notation for Connected Objects [18]). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ the set of all connected objects $O2O' \in O \times O$ is given by $O2O' = O2O \cup \{(o_1, o_2) \in O \times O \mid o_1 \neq o_2 \wedge \exists e \in E \colon \{o_1, o_2\} \subseteq \pi_{obj}(e)\}$. Also, we use $\overline{O2O} = \{(o_1, o_2) \in O \times O \mid (o_1, o_2) \in O2O \vee (o_2, o_1) \in O2O\}$ to describe the set of objects for which at least one direction is part of $O2O'$.*

We call two objects connected if they have an object-to-object relation. We call two types connected if at least one pair of connected objects has one object of each type. The set $CT(L)$ contains one direction of all connected object types.

Definition 3 (Connected Types [18]). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ we reduce a set of objects $O' \subseteq O$ to objects of type $t \in OT$ with the following notation: $O'_{\downarrow t} = \{o \in O' \mid \pi_{type}(o) = t\}$.*
We reduce the connected objects to pairs of type $t_1, t_2 \in OT$ with $\overline{O2O}_{\downarrow t_1, t_2} = \{(o_1, o_2) \in \overline{O2O} \mid o_1 \in O_{\downarrow t_1} \wedge o_2 \in O_{\downarrow t_2}\}$.
The set $\overline{CT(L)} = \{(t_1, t_2) \in OT \times OT \mid \exists o_1 \in O_{\downarrow t_1}, o_2 \in O_{\downarrow t_2} \colon (o_1, o_2) \in \overline{O2O}\}$ is the set of all connected types.
The set $CT(L) \subset OT \times OT$ contains exactly one connection of all connected types such that $\forall t_1 \neq t_2 \in OT : (t_1, t_2) \in \overline{CT(L)} \Rightarrow ((t_1, t_2) \in CT(L) \oplus (t_2, t_1) \in CT(L))$ and selfloops are excluded $\forall t_1 \in OT : (t_1, t_1) \notin CT(L)$.

We assume that an object's lifespan starts with its first recorded event and ends with its last recorded event.

Definition 4 (Object Lifespan Interval). *Given an event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and object* $o \in O$. *The start of the lifespan of object* o *is* $s_L(o) = min(\{t \in \mathbb{U}_{time} \mid \exists_{e \in E} \pi_{obj}(e) = o \wedge \pi_{time}(e) = t\})$. *The end of the lifespan of an object* o *is* $e_L(o) = max(\{t \in \mathbb{U}_{time} \mid \exists_{e \in E} \pi_{obj}(e) = o \wedge \pi_{time}(e) = t\})$. *The lifespan of an object is the interval* $ols(o) = \{t \in \mathbb{U}_{time} \mid s_L(o) \leq t \leq e_L(o)\}$.

Temporal relations between object types group together similar lifespan relations to categorize the lifespan relations of connected objects of two object types. The during (■) relation describes that the lifespans of objects of type A are happening within the lifespan of connected objects of type B if type A is during type B. For example, all types happen during the *Company* type because its lifespan covers all other lifespans. The proceeds (►) relationship describes that the lifespan of objects of type A starts before those of connected objects of type B starts and ends before the ones of type B end if type A proceeds type B. In the running example, this is the case for *Orders* that proceeds the *Items*. The parallel (‖) relations allow for all types of temporal relations and act as a fallback option in case the other relation does not fit.

Definition 5 (Temporal Relations [18]). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and filter parameter* $0 \leq \tau \leq 1$ *the following temporal relations hold for a tuple of types* $(t_1, t_2) \in OT \times OT$ *:*

$(t_1, t_2) \in \blacksquare_{L,\tau}$ *iff* $\frac{|\{(o_1,o_2) \in \overline{O2O}_{\downarrow t1,t2} | s_L(o2) \leq s_L(o1) \wedge e_L(o1) \leq e_L(o2)\}|}{|\overline{O2O}_{\downarrow t1,t2}|} \geq \tau$

$(t_1, t_2) \in \blacksquare^{inv}_{L,\tau}$ *iff* $(t_2, t_1) \in \blacksquare_{L,\tau}$

$(t_1, t_2) \in \blacktriangleright_{L,\tau}$ *iff* $\frac{|\{(o_1,o_2) \in \overline{O2O}_{\downarrow t1,t2} | s_L(o1) < s_L(o2) \wedge e_L(o1) < e_L(o2)|}{|\overline{O2O}_{\downarrow t1,t2}|} \geq \tau \wedge (t_1, t_2) \notin \blacksquare_{L,\tau} \cup \blacksquare^{inv}_{L,\tau}$

$(t_1, t_2) \in \blacktriangleright^{inv}_{L,\tau}$ *iff* $(t_2, t_1) \in \blacktriangleright \wedge (t_1, t_2) \notin (\blacksquare_{L,\tau} \cup \blacksquare^{inv}_{L,\tau})$

$(t_1, t_2) \in \|_{L,\tau}$ *iff* $(t_1, t_2) \notin (\blacksquare_{L,\tau} \cup \blacksquare^{inv}_{L,\tau}) \wedge (t_1, t_2) \notin (\blacktriangleright_{L,\tau} \cup \blacktriangleright^{inv}_{L,\tau})$

The TOTeM model combines the temporal relation with the log cardinality and the event cardinality to describe the relations between object types. Since we focus on the temporal relation for the remainder of the paper, concrete definitions for the log and event cardinality are not listed here but are described in the original paper [18]. TOTeM is a directed graph between connected object types where each edge is labeled with the most precise temporal relation for object pairs of the two types.

Definition 6 (Temporal Object Type Model [18]). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and filter parameter* $0 \leq \tau \leq 1$ *the temporal object type model* $TOTeM_{L,\tau}$ *is a directed graph* $T_{L,\tau} = (OT, CT(L), \pi_{tr}, \pi_{cc}, \pi_{oc})$ *where:*

- $CT(L)$ *is the set of directed edges between connected types;*

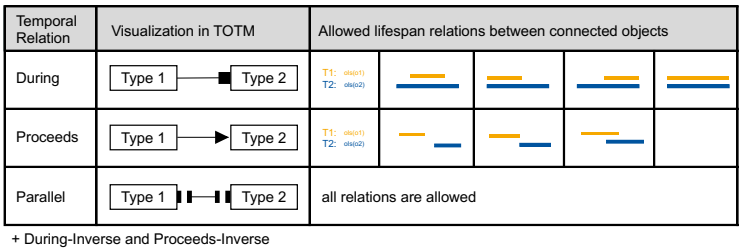

Fig. 3: Figure adapted from [18]. The temporal relation *during*(■), *precedes*(▶), and *parallel*(||) (based on Allen's interval algebra [3]). For each temporal relation, the allowed lifespan relations for connected objects of *Type 1* (in orange) and *Type 2* (in blue) are shown.

- $\pi_{tr} : CT(L) \to \{\blacksquare_{L,\tau}, \blacksquare^{inv}_{L,\tau}, \blacktriangleright_{L,\tau}, \blacktriangleright^{inv}_{l,\tau}, \|_{l,\tau}\}$ *labels each edge with a temporal relation;*
- $\pi_{lc} : CT(L) \to \{1, 0..1, 1..*, 0..*\}$ *labels each edge with a log cardinality that holds (see [18]);*
- $\pi_{ec} : CT(L) \to \{0, 1, 0..1, 1..*, 0..*\}$ *labels each edge with an event cardinality that holds (see [18]).*

We use the notation $T_{\downarrow OT}$ *with* $OT \in \mathbb{U}_{type}$ *to describe the reduction of TOTeM model* T *to object types in* OT*.*

The transitive reduction of the temporal relations of the TOTeM model for the running example can be seen in Figure 4. One can see that all object types have a *during* relation with the *Company* object type. This shows that the lifespan of the company object covers all other object lifespans within the event log. So, for example, *Factory* objects never exist before or after the lifespan of the company object. Also, one can see that *Worker* and *HR* objects have a parallel temporal relation, showing that their lifespans are unordered. So sometimes *HR* employees leave before *Worker* employees that worked together, and sometimes it is the other way around. A more structured lifespan relation is the proceeds relation from *Order* to *Item*. This shows that the order always exists before the connected items appear in the event log. TOTeM models can be discovered from object-centric event logs using the TOTeM miner presented by Liss et al. [18].

4 Process Area Detection

In this section, we present the approach to detect process areas from an object-centric event log. The approach consists of three steps. The first step is mining a TOTeM model for the given event log. The temporal relations in the TOTeM model are then used in the second step to assign object types to levels of the process such that higher-level object types act as resources for lower-level object types. In the third step, connected object types of the same level are combined,

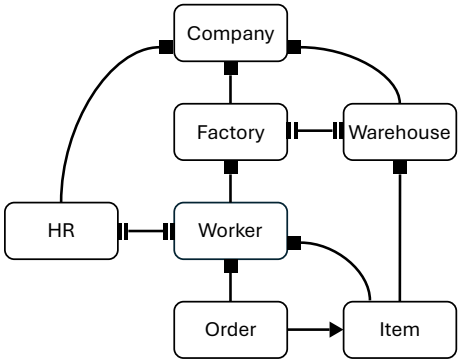

Fig. 4: TOTeM model of the order management running example process, showing the relation between the lifespans of the object types of the process.

and relevant activities for these types are determined to form a process area. This process area describes a process within the organization's system of processes.

The first step, mining a TOTeM model, has been presented by Liss et al. [18]. Thus, detailed descriptions of how to discover TOTeM models can be found in that paper. The TOTeM model describes the temporal, log cardinality, and event cardinality relations between object types in the object-centric event log.

The second step aims to separate object types into levels so that object types are at a higher level than the object types for which they act as resources. This step is based on the assumption that resources have a longer lifespan at organizations than the object that uses this resource. For example, *Factories* outlive multiple *Workers* who themselves outlive multiple *Orders*. The TOTeM model captures this behavior with the during (■) relation. There type A is during type B if they are connected, and the lifespan of objects of type A always occurs during the lifespan of connected objects of type B. Therefore, we want type A to have a lower level than type B because, based on our assumption, type B acts as a resource for type A. For the running example, we see that, for example, for the previously mentioned types *Factories*, *Workers*, and *Orders*. There, orders are during workers, and workers are during factories, as one can see in the TOTeM model for the running example in Figure 4.

However, having higher-level types acting as resources for lower-level types is not the only requirement for the level assignment. Therefore, the requirement of having higher level types acting as resources for lower level types can be reformulated as follows: types should be on a lower level than types they have a during relationship to. To keep the assignment as compact as possible, the overall distance between connected object types should be as low as possible for the level assignment. This also makes types with multiple possible levels end up on levels with the types they share the most connections with. This layer assignment problem, given an object-centric event log and a TOTeM model, can be described as the Integer Linear Program (ILP) defined in Definition 7.

Definition 7 (Resource Layer Assignment ILP). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$ *and filter parameter* $0 \leq \tau \leq 1$*, the resource layer assignment ILP consists of the following equations given small* $\epsilon \in \mathbb{R}^+$*:*

$$
\begin{aligned}
&\textit{minimize} && \sum_{\{ot_1, ot_2\} \in \blacktriangleright_{L,\tau} \cup \|_{L,\tau}} |level(ot_2) - level(ot_1)| + \\
& && 0.5 \cdot \sum_{(ot_1, ot_2) \in \blacksquare_{L,\tau}} level(ot_2) - level(ot_1) + \\
& && \epsilon \cdot \sum_{ot \in OT} level(ot) \\
&\textit{subject to} && \\
& && level(ot_2) - level(ot_1) \geq 1 && \forall (ot_1, ot_2) \in \blacksquare_{L,\tau} \\
& && level \in OT \rightarrow \mathbb{N}_0
\end{aligned}
$$

The variables of the linear program $level(ot)$ describe the level of object type ot. As described before, the distance between connected object types should be as low as possible while keeping the requirements intact. The last row of the requirements ensures that the levels are discrete numbers and only positive levels (including 0) are assigned. The first requirement in the ILP ensures that types are placed higher than those they act as a resource for, so those that have a during relation to them. The minimization function is the absolute distance value of all connected object types to keep the layer assignment compact. Since the level of types that happen during another type is always lower, their absolute distance can be replaced with a simple subtraction since it is known which level is higher than the other. This explains the second term in the minimization function. The third term in the minimization function ensures that the level assignment starts with 0. Note that the absolute function is not linear and, therefore, needs to be reformulated to solve the ILP. There are standard ways to transform absolute functions by introducing additional variables to get a resulting linear ILP [15]. Since the during relation is transitive and the TOTeM model has no self-loops, the TOTeM model is an acyclic graph when reduced to the during relations. For acyclic graphs, the layer assignment problem always has a solution, so the ILP will always return one of the potentially many optimal solutions [11].

Figure 5 shows the resulting optimal layer assignment for the object types of the running example. One can see that the requirement regarding the during relation is satisfied, and all types that happen during another one are placed on a lower level. For example, the *Order* is lower than the *Worker*, which is lower than the *Factory*, which is lower than the *Company*. This does not mean that each type is placed directly below the types they have a during relation with. This can be seen for example with the *HR* type which is placed on level 1 although it could have also been placed on level 2 and level 0 (indicated by the dotted boxes in Figure 5). But since the HR type interacts with the workers, we prefer them to be on the same level. This happens because we weight the minimization of parallel and proceeds relations higher than the minimization of the during relation in the minimization function.

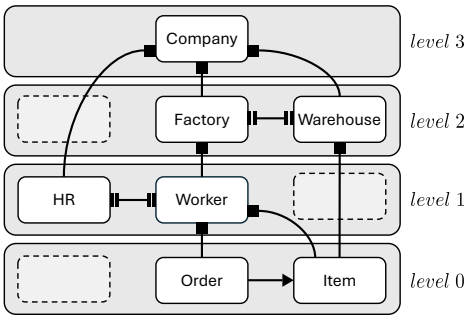

Fig. 5: The optimal layer assignment for the object types of the running example. The dashed places indicate alternative suboptimal levels for *HR* and *Warehouse*.

Given the layer assignment, we can derive levels, where each level contains the object types of that level and relevant activities of that level. Activities are assigned to the lowest level, which contains an object type that appears in the activities' events. Thus, each level focuses on the activities relevant to that level, and the process for higher-level types is not cluttered with activities that use them as resources.

Definition 8 (Process Levels). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$*, the filter parameter* $0 \leq \tau \leq 1$*, and an optimal solution of the resource layer assignment ILP the object types* $l_{ot}(i) \subseteq OT$ *and activities* $l_{act}(i) \subseteq \mathbb{U}_{act}$ *for the level* $i \in \mathbb{N}_0$ *of the process are defined as:*

$$l_{ot}(i) = \{ot \in OT \mid level(ot) = i\}$$
$$l_{act}(i) = min(\{level(\pi_{type}(o)) | o \in O \wedge e \in E \wedge o \in \pi_{obj}(e) \wedge \pi_{act}(e) = a\})$$

For example, the object types of the first level ($l_{ot}(1)$) for the running example are *Order* and *Item*. Since it is the lowest level, all the activities they are involved in are assigned to this level, so for example, *create order*, *produce item*, *package order*, and *send order*. For the second level, the object types are *HR* and *Worker*. Not all activities that the *Worker* is involved in are added here because some of them are already assigned to level 0. For example, the workers are involved in the *produce item* activity. However, that is already assigned to level 0, which makes sense because the worker only acts as a resource for this activity.

Process areas describe a set of object types and activities which can be additionally marked with object types that act as a resource for certain selected activities. A general definition of process areas is given in Definition 9. The TOTeM-based process areas, which are based on the levels, are described in Definition 10.

Definition 9 (Process Area). *Given event log* $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$*, a process area* $pa = (OT', AC', R) \in \mathcal{P}(OT) \times \mathcal{P}(\mathbb{U}_{act}) \times (\mathbb{U}_{act} \rightarrow OT)$ *selects object types* OT' *activities* $AC' \subseteq image(\pi_{act})$ *and* R *maps these*

activities to object types that act as a resource for the given activity but do not belong to the set of selected object types: $image(R) \cap OT' = \emptyset$. The set of all possible process areas for L is noted as PA_L.

The TOTeM-based process areas are detected by splitting process levels into separate process areas for each group of types that form a connected component when restricting the TOTeM model to object types of that level. This ensures that only those types end up in the same process area that actually interact with each other. Also, each activity in a process area is annotated with a set of connected types from higher levels, which act as a resource for this activity. This represents the resource involvement in a less cluttered way.

Definition 10 (TOTeM-based Process Areas). *Given event log $L = (E, O, OT, O2O, \pi_{act}, \pi_{obj}, \pi_{time})$, the filter parameter $0 \leq \tau \leq 1$, and the process level object types $l_{ot}(i)$ and activities $l_{act}(i)$ for process level $i \in \mathbb{N}_0$ the set of TOTeM-based process areas $PAS \subseteq PA_L$ such that for all $(OT', AC', R) \in PAS$:*

$$\begin{aligned}\exists i \in \mathbb{N}_0 \colon\ & OT' \subseteq l_{ot}(i) \wedge OT' \text{ is connected component in } T_{\downarrow OT'} \wedge \\ & AC' \subseteq l_{act}(i) \cap \{\pi_{act}(e) \in \mathbb{U}_{act} \mid e \in E \wedge \exists o \in O \colon \pi_{type}(o) \in OT' \wedge \\ & o \in \pi_{obj}(e)\} \\ & \forall a_1 \in AC' \colon R(a_1) = (\textstyle\bigcup_{j \in \mathbb{N}_0, i<j} l_{ot}(j)) \cap \{ot \in OT \mid \exists e \in E, o \in O \colon \\ & \pi_{act}(e) = a_1 \wedge o \in \pi_{obj}(e) \wedge \pi_{type}(o) = ot\}\end{aligned}$$

Resource mapping R maps the activities of that process area to their resources from higher levels. Each object type is part of exactly one process area but can act as a resource for multiple ones.

By applying the above-presented concepts to an object-centric event log, one can compute TOTeM-based process areas for a given log. The algorithm first computes a TOTeM model, then creates and solves the resource layer assignment ILP, to create TOTeM-based process areas is described in Algorithm 1.

Algorithm 1 TOTeM-based Process Area Detection

Require: : Object-Centric event log L; TOTeM support threshold τ.

#Compute a TOTeM model using the totem_miner from [18]
$T_{L,\tau} \leftarrow totem\_miner(L, \tau)$
#Transform TOTeM to resource layer assignment ILP
$levelILP \leftarrow rla\_ILP(L, T_{L,\tau})$
#Solve ILP and compute object types and activities for each level $i \in \mathbb{N}_0$
$level \leftarrow solve(levelILP)$
$l_{ot}(i) \leftarrow \{ot \in OT \mid level(ot) = i\}$
$l_{act}(i) \leftarrow min(\{level(\pi_{type}(o)) | o \in O \wedge e \in E \wedge o \in \pi_{obj}(e) \wedge \pi_{act}(e) = a\})$
#Return TOTeM-based process areas
$return\ PAS(T_{L,\tau}, l_{ot}, l_{act})$

Figure 1 shows the process areas of the running example. The object types of the process areas are indicated by the blue groups in the TOTeM model, and the activities and their resource mapping are shown as an object-centric Petri net per process area. We annotated each activity with the types that act as resources for the process area and participate in the activity. The detected process areas match the types of processes of the running example production company. The order management area, the employee area, and the company lifecycle area are clearly separated and correctly detected.

5 Evaluation

This section describes the qualitative and quantitative evaluation that we performed using the publicly accessible implementation of our approach to investigate the feasibility of our approach on consumer hardware with publicly accessible event logs.

5.1 Qualitative Evaluation

The qualitative evaluation was done using a publicly accessible object-centric event log that describes an organization that handles container logistics [16]. We applied our process area detection algorithm with $\tau = 0.9$, as described in Section 4 on the object-centric event log in the OCEL 2.0 format. The event log contains 35.761 events that originate from 14 activities. It involves 14.013 objects from 7 object types.

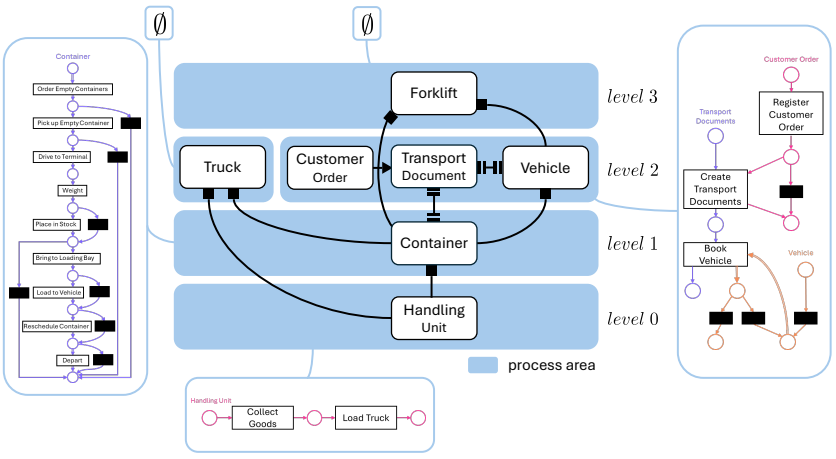

Fig. 6: Process areas for the container logistic log [16] and object-centric Petri nets describing the behavior of the process areas.

The resulting TOTeM model, the assignment of object types to levels, and the final groups of object types that are process areas for the container logistics

process are shown in Figure 6. Also, for each process area, we show the object-centric Petri net, discovered using ocpa [1].

This visualization allows to get a compact overview of the processes running in the organization and shows what roles each object type takes. For example, for the *forklifts*, one can see that they have no behavior on their own (indicated by the empty object-centric Petri net of that process area visualized with $\emptyset$). This shows that forklifts in the organization purely act as a resource for lower-level process areas like container handling (level 1 in Figure 6). These containers have a clear sequential process, as shown on the left side of Figure 6. Another interesting insight can be gained from the process area at level 2 that contains the *Customer Order*, the *Transport Document*, and the *Vehicle* object type. For this process area, the resulting object-centric Petri net clearly shows the interactions of the three involved object types with the disturbance of resources. The object-centric Petri net on the right side of Figure 6 shows that customer orders trigger the creation of *Transport Documents*. The lifespan of the vehicles occurs in parallel to the *Transport Document*s, but multiple vehicles can be involved in the book vehicle event. This shows that the detected process areas can keep relevant object type interactions within a process execution and separate resources that would hinder the understandability of the models.

5.2 Quantitative Evaluation

For the qualitative evaluation, we investigated the runtime of our publicly accessible implementation for the process area detection on nine public object-centric event logs. The event logs are in the OCEL format (both version 1.0 and 2.0). Our implementation, the code for the evaluation, and links to the used event logs can all be found in the public repository on GitHub. The machine used for the evaluation has a 64-bit operating system, 16 GB RAM, and a 3.2 GHz AMD processor. For each log, the resulting number of levels, number of process areas, the average number of types per process area, and the average number of activities per process area are presented in Table 2. We measured the runtime for

Table 2: The number of levels, the number of process areas, the average number of types per process area, and the average number of activities for all event logs.

Log	# levels	#process areas	avg. # types	avg. # activities
Container Logistics	4	5	1.4	2.8
Github PM4Py	2	2	1.5	10
O2C	7	15	1.27	1.53
Order Management	3	4	1.5	2.75
P2P	4	6	1.17	5.17
P2P-normal	4	4	1.25	2.25
Recruiting	3	3	2	5.33
Transfer	2	2	2.5	1.5
Windows	3	3	2	112.33

each of the nine logs ten times and took the average of those ten measurements. The results of each log are plotted over the number of events and the number of objects in Figure 7. In addition, both plots show a regression line indicating that the runtime grows linearly with the number of objects and events.

The results show that the approach's performance makes it applicable to detecting process areas on consumer hardware using current publicly accessible logs. The longest measured run-time was 23.34 for the *O2C* event log. Within the runtime, the most time (about 99% of the total runtime) is spent mining the TOTeM model. Creating the ILP for the mined TOTeM model and also solving that is quick in comparison. This is also expected since the TOTeM miner has to go over all events and objects individually, whereas the rest of the approach operates on the aggregated information on the object-type level. All in all, the results of the qualitative evaluation show the applicability of the approach for object-centric event logs similar in size to current publicly accessible logs.

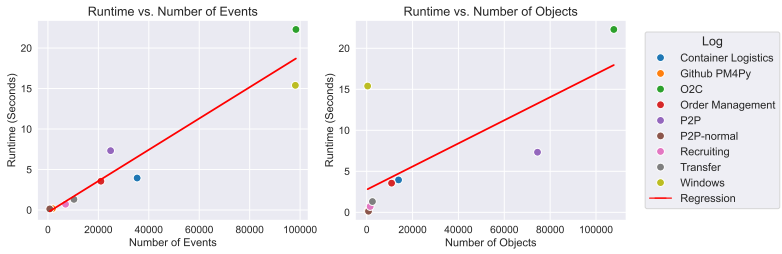

Fig. 7: Runtime evaluation on multiple public object-centric event logs.

6 Conclusion

This work introduces object-centric process areas and an approach to discovering them from an object-centric event log by presenting three contributions. First, we define an algorithm to determine multiple levels of resources within the organization. We utilize the temporal relation discovered in TOTeM models together with the assumptions that types have a during relation to connected resource object types. We propose a way to detect process areas using this multilevel resource detection. Second, we provide a publicly accessible implementation in Python for the complete approach. Finally, we evaluated the approach qualitatively and qualitatively. The qualitative evaluation shows that the approach can detect relevant process areas and uncover the connections between higher-level resources and lower and higher-level processes. The quantitative evaluation shows that, based on our implementation's runtime, process areas can be computed in reasonable time on consumer hardware.

Of course, the approach comes with certain assumptions and limitations, affecting the effectiveness for certain use cases. It is assumed that objects that

act as a resource have a longer lifetime. If this is not the case, TOTeM-based process areas may not yield meaningful results. The concept of process areas is still relevant for these use-cases, but other mining approaches with other assumptions should be investigated as future research to support these scenarios. Also, process areas focus on de-cluttering the non-resource workflows. If one is interested in the workflow of resource objects themselves, like the *Forklift* behavior in Figure 6, one can still use the level assignment to detect object types that act as resources, but should combine this information with manual filtering for investigating the resource behavior. In general, process areas are a fully automated initial grouping of object-centric behavior, and they can always be fine-tuned further using manual filtering by domain experts.

Process areas motivate future research that investigates the relations between resource objects and non-resource objects within the process area and between them. Future research towards lifting the assumption that the complete lifespan of objects is present in the log could make this approach applicable for streaming settings. Also, an empirical evaluation of the understandability of the mined process with and without process areas represents interesting future research. The clear notion of multiple levels of object types that act as resources for lower levels enables a more thorough investigation of resource patterns and structures in object-centric process mining.

Acknowledgment The project on which this work is based upon was funded by the German Federal Ministry of Education and Research under grant number 01IF25011. The responsibility for the content of this publication lies with the authors.

References

1. Jan Niklas Adams, Gyunam Park, and Wil M. P. van der Aalst. ocpa: A python library for object-centric process analysis. *Softw. Impacts*, 14:100438, 2022.
2. Jan Niklas Adams, Daniel Schuster, Seth Schmitz, Günther Schuh, and Wil M. P. van der Aalst. Defining cases and variants for object-centric event data. In *4th International Conference on Process Mining, ICPM*, pages 128–135. IEEE, 2022.
3. James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
4. Nuha Alshuqayran, Nour Ali, and Roger Evans. A systematic mapping study in microservice architecture. In *SOCA*, pages 44–51, 2016.
5. Bianka Bakullari and Wil M. P. van der Aalst. High-level event mining: A framework. In *ICPM*, pages 136–143. IEEE, 2022.
6. Alessandro Berti and et. al. OCEL (object-centric event log) 2.0 specification. *CoRR*, abs/2403.01975, 2024.
7. Marius Breitmayer, Lisa Arnold, and Manfred Reichert. Enabling conformance checking for object lifecycle processes. In *Research Challenges in Information Science, RCIS*, volume 446 of *LNBIP*, pages 124–141. Springer, 2022.
8. Axel Kjeld Fjelrad Christfort, Andrey Rivkin, Dirk Fahland, Thomas T. Hildebrandt, and Tijs Slaats. Discovery of object-centric declarative models. In *ICPM*, pages 121–128. IEEE, 2024.

9. Dirk Fahland. Process mining over multiple behavioral dimensions with event knowledge graphs. In *Process Mining Handbook*, volume 448 of *LNBIP*, pages 274–319. Springer, 2022.
10. Riccardo Galanti, Massimiliano de Leoni, Nicolò Navarin, and Alan Marazzi. Object-centric process predictive analytics. *Expert Syst. Appl.*, 213(Part):119173, 2023.
11. Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo. A technique for drawing directed graphs. *IEEE Trans. Software Eng.*, 19(3):214–230, 1993.
12. Wissam Gherissi, Joyce El Haddad, and Daniela Grigori. Object-centric predictive process monitoring. In *Service-Oriented Computing - ICSOC 2022 Workshops*, volume 13821 of *LNCS*, pages 27–39. Springer, 2022.
13. Alessandro Gianola, Marco Montali, and Sarah Winkler. Object-centric conformance alignments with synchronization. In *CAiSE*, volume 14663 of *LNCS*, pages 3–19. Springer, 2024.
14. Gregor Gößler and Joseph Sifakis. Composition for component-based modeling. *Sci. Comput. Program.*, 55(1-3):161–183, 2005.
15. Jack E. Graver. On the foundations of linear and integer linear programming I. *Math. Program.*, 9(1):207–226, 1975.
16. Benedikt Knopp and Nina Graves. Container logistics object-centric event log, October 2023.
17. Lukas Liss, Jan Niklas Adams, and Wil M. P. van der Aalst. Object-centric alignments. In *ER*, volume 14320 of *LNCS*, pages 201–219. Springer, 2023.
18. Lukas Liss, Jan Niklas Adams, and Wil M. P. van der Aalst. Totem: Temporal object type model for object-centric process mining. In *BPM Forum*, volume 526 of *LNBIP*, pages 107–123. Springer, 2024.
19. Kalle Lyytinen, Barbara Weber, Markus C. Becker, and Brian T. Pentland. Digital twins of organization: implications for organization design. *Journal of Organization Design*, 13(3):77–93, September 2024.
20. Viki Peeva, Marvin Porsil, and Wil M. P. van der Aalst. Object-centric local process models, 2024.
21. Manfred Reichert and Barbara Weber. *Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies*. Springer, 2012.
22. Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.
23. Wil M. P. van der Aalst. Object-centric process mining: Dealing with divergence and convergence in event data. In *Software Engineering and Formal Methods, SEFM*, volume 11724 of *LNCS*, pages 3–25. Springer, 2019.
24. Wil M. P. van der Aalst and Alessandro Berti. Discovering object-centric petri nets. *Fundam. Informaticae*, 175(1-4):1–40, 2020.
25. Jan Niklas van Detten, Pol Schumacher, and Sander J. J. Leemans. Discovering compact, live and identifier-sound object-centric process models. In *ICPM*, pages 113–120. IEEE, 2024.
26. Jan Niklas van Detten, Pol Schumacher, and Sander J. J. Leemans. A framework for advanced case notions in object-centric process mining. In *ICPM Workshops*, 2024.
27. Sebastiaan J. van Zelst, Felix Mannhardt, Massimiliano de Leoni, and Agnes Koschmider. Event abstraction in process mining: literature review and taxonomy. *Granular Computing*, 6(3):719–736, July 2021.