Extending Process Intelligence with Quantity-Related Process Mining

Nina Graves(✉), Tobias Brockhoff, István Koren, and Wil M. P. van der Aalst

Chair of Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
{graves,brockhoff,koren,wvdaalst}@pads.rwth-aachen.de

Abstract. Process mining uses data logged during the execution of processes to understand, analyse, and improve processes. Logistics process management and optimisation are highly relevant for the industry, as they are crucial to the business' operations but not intrinsically value-adding. Despite the advantages of applying process mining to logistics processes, its full potential can not yet be leveraged. Current process mining techniques assume that an event's execution solely depends on its associated identifiable objects, their attributes, relationships, and previously executed events. However, in logistics processes, *counts* of items, which may not be uniquely identifiable, play a crucial role. For instance, a replenishment order is triggered when the stock level falls below a threshold, or a second shipment is dispatched if not all ordered items are available during the first shipment. This work proposes a framework that integrates the concept of a *quantity state* based on properties derived from common logistics processes. We introduce extensions to object-centric event logs and object-centric Petri nets that include such counts of items. We show the feasibility of detecting the quantity state from the proposed event log class and demonstrate its capability to convey quantity dependencies using a Python-based implementation.

Keywords: object-centric process mining · logistics · event log

1 Introduction

Recently, the need for increasing transparency in supply chains has become more pronounced, and process mining (PM) is considered a strong contester to provide it [7]. However, PM is not well established in the domain of logistics [5] as not all relevant data can be added into the event log format explicitly [8] and techniques do not support a process notion beyond independent workflows [15,16]. Due to their high industrial relevance, inventory management processes (IMPs) are widely studied and classified in the fields of logistics and supply chains [3]. In several of the common strategies for their handling, the execution of an event depends on a particular *count* of items, e.g., products in a warehouse or unreserved products in a planning system. Such counts can be affected by the execution of the workflow of specific objects but are not tied to their lifecycles –

A. Delgado and T. Slaats (Eds.): ICPM 2024 Workshops, LNBIP 533, pp. 337–349, 2025.
https://doi.org/10.1007/978-3-031-82225-4_25

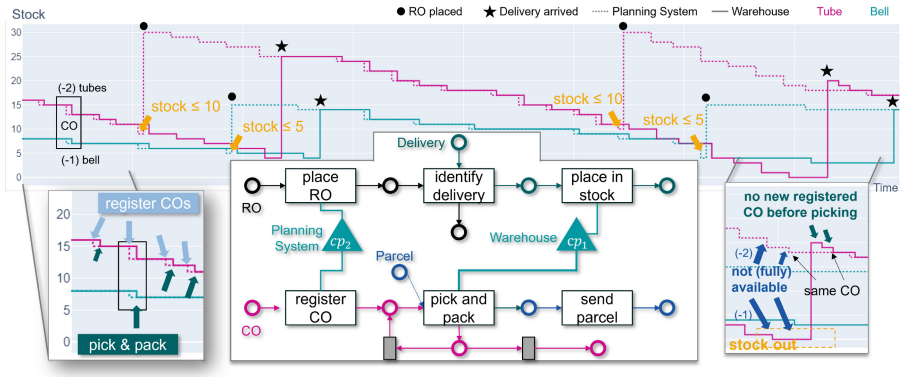

Fig. 1. Example IMP with two products, tubes and bells, and two different counts, cp_1 and cp_2, included in the extended object-centric Petri net (centre). The chart (top) shows the development of the stock levels in the warehouse (solid) and the planning system (dotted) over time. Dotted decreases occur when COs are registered, and solid decreases when products are removed (left). ROs are placed when ten or fewer tubes or five or fewer bells are available in the planning system. If COs arrive that cannot be fulfilled, all available products are removed from the warehouse, and the removal is repeated after the delivery arrives (right).

individual process executions are not independent. Consider, for example, the IMP depicted in Fig. 1, in which a replenishment order (RO) is placed when the number of products in the company's planning system (cp_2) falls to ten tubes or five bells or below. After the order is placed, the in-transit products are registered in the planning system, and when the delivery arrives, the products are placed into the company warehouse (cp_1). Whenever a customer order (CO) arrives, the number of ordered products is removed from the planning system when the order is registered. All available and demanded products are then picked from the warehouse and sent to the customer. If the full CO cannot be fulfilled because the ordered products are not available in the warehouse, a second parcel with the missing products is sent. Current PM techniques implicitly assume that (1) the control flow only depends on the current state of identifiable objects, i.e., object attributes and previously executed activities [1], and (2) activities and object types are the only relevant entities which are represented by uniquely identifiable events and objects in the event log [14]. These assumptions do not hold for IMPs, as the execution of an event can also depend on counts of products without unique identifiers (not every pen or cup has an ID).

This work introduces a basic framework for quantity-related process mining (QRPM) by extending object-centric process mining (OCPM) to allow for the additional consideration of a *quantity state* referring to such counts of items. Motivated by IMPs, we derive generic properties for quantity-related processes to extend object-centric event logs, derive the quantity state from such a log, and include the concept of item counts into process models and discovery. Based

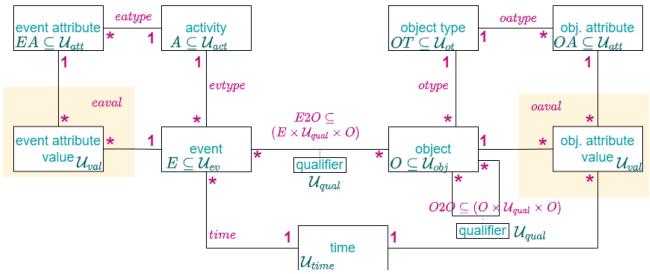

Fig. 2. Meta Model of the Object-Centric Event Log Standard OCEL 2.0 [1].

on the demonstration and discussion of a use case, we conclude the framework's capability of conveying quantity-related behaviour and its suitability as a baseline for further research for PM techniques supporting logistics processes. The remainder of this work is structured as follows: Sect. 2 presents the preliminaries before the framework is introduced in Sect. 3. Section 4 describes the implementation and case study. In Sect. 5, we give an overview of related work before concluding this work in Sect. 6.

2 Preliminaries

A partial function from a set X to a set Y is denoted $f : X \nrightarrow Y$ where $dom(f) \subseteq X$ refers to the function's domain and $rng(f) \subseteq Y$ its range. We denote a total function as $g : X \rightarrow Y$ where $dom(f) = X$. We assume the set-theoretic definition of functions being a relation over tuples, e.g. $g = \{(x_1, f(x_1)), ..., (x_n, f(x_n))\}$ For any finite set A, $|A|$ denotes the number of elements in the set, $\mathcal{P}(A)$ its powerset, and $\mathcal{B}(A)$ the set of all multisets over A. A multiset $b : A \rightarrow \mathbb{N}$ assigns a natural number to every element of a set, $b_1 = [\,]$, $b_2 = [x, y^2, z^3]$, and $b_3 = [y, z^2]$ with $b_1, b_2, b_3 \in \mathcal{B}(A)$ are examples for multisets over a set $A = \{x, y, z\}$. The standard set operators are extended to multisets; $x \in b_2$; $x \notin b_3, b_1$; $b_2 \uplus b_3 = [x, y^3, z^5]$; and $b_2 \cap b_3 = b_3$.

We introduce the following universes and define an Object-Centric Event Log (OCEL) according to the current OCEL 2.0 standard [1]: $\mathcal{U}_{act}$ for activities (i.e., event types), $\mathcal{U}_{ev}$ for events, $\mathcal{U}_{ot}$ for object types, $\mathcal{U}_{obj}$ for objects, $\mathcal{U}_{att}$ for attributes, $\mathcal{U}_{val}$ for attribute values, $\mathcal{U}_{qual}$ for qualifiers, and $\mathcal{U}_{time}$ with $\{0, \infty\} \subseteq \mathcal{U}_{time}$ for time.

Definition 1 (OCEL). *An object-centric event log is a tuple* $L_{oc} = (E, O, EA, OA, act, otype, eatype, oatype, eaval, oaval, time, E2O, O2O)$ *where* $E \subseteq \mathcal{U}_{ev}$ *is a set of events,* $O \subseteq \mathcal{U}_{obj}$ *is a set of objects,* $EA, OA \subseteq \mathcal{U}_{att}$ *are the sets of event and object attributes,* $act : E \rightarrow \mathcal{U}_{act}$ *and* $otype : O \rightarrow \mathcal{U}_{ot}$ *map objects and events to types,* $eatype : EA \rightarrow \mathcal{U}_{act}$ *and* $oatype : OA \rightarrow \mathcal{U}_{ot}$ *map attributes to types,* $eaval : (E \times EA) \nrightarrow \mathcal{U}_{val}$ *and* $oaval : (O \times OA \times \mathcal{U}_{time}) \nrightarrow \mathcal{U}_{val}$ *partially assign values to event and object attributes,* $time : E \rightarrow \mathcal{U}_{time}$

assign timestamps to events, $E2O \subseteq (E \times \mathcal{U}_{qual} \times O)$ are qualified event-to-object relations, and $O2O \subseteq (O \times \mathcal{U}_{qual} \times O)$ are qualified object-to-object relations, such that $dom(eaval) \subseteq \{(E, ea) \in E \times EA \mid act(e) = eatype(ea)\}$ only attributes assigned to an event's activity have values, and $dom(oaval) \subseteq \{(o, oa, tm) \in O \times EA \times \mathcal{U}_{time} \mid otype(o) = oatype(oa)\}$ only attributes of the object's type have a value.

Figure 2 shows the meta model and the mentioned functions for the current OCEL 2.0 standard. Object-Centric Petri Nets (OCPNs) model the joint workflows of multiple object types, based on labelled Petri nets, $N = (P, T, F, l)$, defined in the usual way. As introduced in [14], they are defined as a labelled Petri net, a mapping of object types to places, and a set of variable arcs. Whereas normal arcs indicate the consumption/production of a single token, variable arcs merely indicate that the number of tokens is not always necessarily one.

Definition 2 (Object-Centric Petri net). *An object-centric Petri net is a tuple $OCPN = (N, pt, F_{var})$ with $N = (P, T, F, l)$ a labelled Petri net, $pt : P \to OT$ maps places to object types, and $F_{var} \subseteq F$ is the subset of variable arcs.*

An OCPN's marking is a multiset of tokens, $m_o \in M_o = \mathcal{B}(P \times O)$; a token being the combination of a place and an object. An *execution binding* (t, b_o) refers to a transition and an object binding function specifying the consumed and produced tokens. An execution binding is only enabled if the current marking includes all to-be-consumed tokens.

3 Quantity-Related Process Mining

This work offers a methodological foundation for PM in processes where the control flow depends on the number of (unidentifiable) entities and the state of objects. As IMPs are a well-defined processes in which this dependency is very explicit, we base our initial framework on an abstraction of IMPs. An *item* is an entity of a particular *item type*, and two items of the same type are considered interchangeable; the process merely depends on their count, not their identities, e.g., tubes and bells in the example process. Similar to [6], we define *counters*, which associate a signed integer with item types, e.g., to denote additions or removals of items from warehouses.

Definition 3 (Counter, Item quantity). *Let $I \subseteq \mathcal{U}_{it}$ be a finite subset of item types from the universe of item types. A counter $c : I \to \mathbb{Z}$ is a function that maps each item type $it \in I$ to an item quantity. The set of all possible counters over I is denoted $\mathcal{I}(I) = I \to \mathbb{Z}$. For counters $c_1, c_2 \in \mathcal{I}(I)$, their composition $c_1 \oplus c_2$ is defined element-wise with $(c_1 \oplus c_2)(it) = c_1(it) + c_2(it)$.*

We write $c_1 = [\![\,]\!]$, $c_2 = [\![x, y^{-1}, y^{-1}, z, z, z]\!] = [\![x, y^{-2}, z^3]\!]$, and $c_3 = [\![x^2, y]\!]$ as examples for counters over a set of item types $I = \{x, y, z\}$. We denote the set of all *active* item types in a counter as $set(c) = \{it \in dom(c) \mid c(it) \neq 0\}$. The composition of a multiset of counters $Q \subseteq \mathcal{B}(\mathcal{I}(I))$ is denoted $c = \bigoplus Q$.

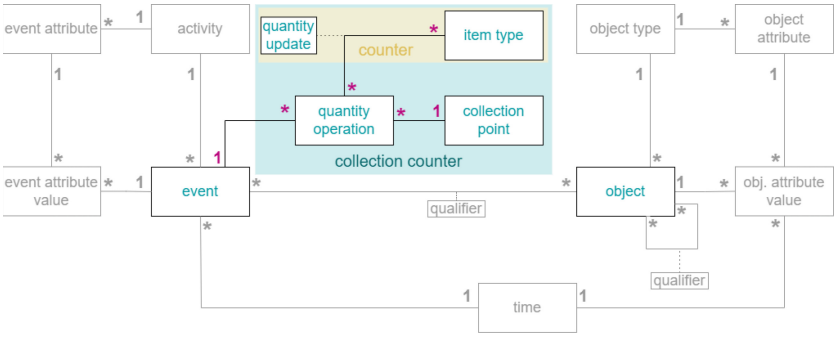

Fig. 3. Meta Model for Quantity Event Logs (QELs).

Collection points (cps) are entities that refer to a particular count of items, e.g., a warehouse. The counts of different item types associated with a cp describe its *item level*, e.g., the item level of the warehouse at the beginning of the example process $[\![tube^{16}, bell^{8}]\!]$. A collection counter maps every cp in a set to a counter.

Definition 4 (Collection Counter). *Let $CP \subseteq \mathcal{U}_{cp}$ be a finite subset of the universe of cps and $I \subseteq \mathcal{U}_{it}$ a finite set of item types. A collection counter is a function $cc : CP \to \mathcal{I}(I)$, and $\mathcal{C}(CP, I) = CP \to \mathcal{I}(I)$ the set of all collection counters over CP and I. The composition of collection counters is $cc_3 = cc_1 \oplus cc_2$ with $cc_3(cp) = cc_1(cp) \oplus cc_2(cp)$ for all $cp \in CP$ for any two collection counters $cc_1, cc_2 \in \mathcal{C}(CP, I)$.*

An example for a collection counter over a set of cps $CP = \{cp_1, cp_2\}$ for the item types $I = \{x, y, z\}$ is $cc_1 = \{(cp_1, [\![x^{-1}, y^{-2}, z^{2}]\!]), (cp_2, [\![x, y]\!])\}$. Composing cc_1 and $cc_2 = \{(cp_1, [\![x, y, z]\!]), (cp_2, [\![x, y^{-2}, z^{-2}]\!])\}$ leads to $cc_1 \oplus cc_2 = \{(cp_1, [\![y^{-1}, z^{3}]\!]), (cp_2, [\![x^{2}, y^{-1}, z^{-2}]\!])\}$. Any $cc \in \mathcal{C}(CP, I)$ is considered *active* if the joint set of active item types is non-empty, i.e., $|\bigcup_{c \in rng(cc)} set(c)| > 0$.

Based on the typology of IMPs presented in [3], we derive three properties characterising quantity-related processes: **(P1) Quantity changes**: An event can change item levels of one or more cps. The affected item levels and the number of items added/removed can differ among events of the same activity. For example, every execution of "place in stock" adds all delivered items to the warehouse, but not all deliveries include the same (number of) items. **(P2) Quantity-dependency for executions**: The execution of an event can depend on the item levels of one or more cps; e.g., tubes are only reordered if the stock level is 5 or below. **(P3) Quantity-dependency for operations**: The number of items added to or removed from a cp in an event can depend on current item levels, e.g., we cannot pick all ordered items if they are not available, so we pick fewer. These three principles highlight the need for PM techniques to explicitly consider a *quantity state*, i.e., the item levels of cps. In logistics, the integration of relevant data into the event log format is a known problem [5,8]: As item types do not require an identifier and the "lifecycle" of a cp cannot be described

Table 1. Data describing active quantity operations for the example process.

event data (for comprehension)			active qops		
eid	activity	timestamp	cp	tube	bell
init	initial item level		cp2	11	7
init	initial item level		cp1	12	7
21	pick and pack	12.10.2019 12:21	cp1	−1	
22	send parcel	12.10.2019 13:16			
23	register CO	12.10.2019 13:56	cp2	−2	−1
24	place RO	12.10.2019 14:01	cp2	21	
25	pick and pack	12.10.2019 14:11	cp1	−2	−1
26	send parcel	12.10.2019 15:19			

by a workflow, their representation as objects is unsuitable. As attributes, they would not be distinct, i.e., automated analyses are impossible. To encompass the inclusion of known collection points and item types, we introduce *quantity event logs* (QELs), which extend OCELs (see Fig. 3). In a QEL, events are connected to *quantity operations*, which describe a change to a cp's item level with *quantity updates* as the number of items added/removed of a specific item type. Consider the example depicted in Fig. 1: when the second CO is registered (black box on the left), the quantity operation to the planning system is $[\![tubes^{-2}, bell^{-1}]\!]$ – the "steps" in each line represent the quantity updates. To accurately and definitively derive the quantity state from a QEL, we require an entry providing the initial item levels of every cp and a strict total order among the events.

Definition 5 (Quantity Event Log). *A quantity event log is a tuple* $QEL = (L_{oc}, I, CP, eqty, \prec_e)$, *with* $L_{oc} = (E, O, EA, OA, act, otype, eatype, oatype, eval, oval, time, E2O, O2O)$ *an OCEL,* $I \subseteq \mathcal{U}_{it}$ *a set of item types,* $CP \subseteq \mathcal{U}_{cp}$ *a set of collection points,* $eqty : (E \cup \{\blacktriangleright\}) \to \mathcal{C}(CP, I)$ *a mapping of events to quantity operations for all collection points, and* $\prec_e \subseteq (E \times E)$ *the strict total order among the events, such that* $\blacktriangleright \notin E$.

We consider an event $e \in E$ to be active if the corresponding collection counter is active, i.e., $rng(eqty(e)) \neq \{[\![]\!]\}$. Table 1 shows example data containing the active quantity operations associated with events to be added to an event log. The column's activity and timestamp are included to ease comprehension. Assuming all relevant quantity operations are logged in the QEL, the quantity state before every event can be determined by composing all prior quantity operations and the initial item level.

Definition 6 (Quantity State). *Let* $QEL = (L_{oc}, I, CP, eqty, \prec_e)$ *be a quantity event log referring to the set of events* $E \subseteq \mathcal{U}_{ev}$. *The function* $qstate : E \to \mathcal{C}(CP, I)$ *describes the quantity state at the execution of* $e \in E$:

$$qstate(e) = \bigoplus [eqty(\blacktriangleright)] \uplus [eqty(e') \mid e' \prec_e e]$$

Equivalently, the quantity state after an event's execution can be determined by adding the event's quantity operation, $post(qstate(e)) = qstate(e) \oplus eqty(e)$. Figure 1 depicts the quantity state of two collection points (types of lines) of the events in QEL describing the example process ordered by time (x-axis).

To model quantity-related processes, we introduce *Quantity Nets* (q-nets), which convey an activity's ability to impact the system's quantity state. In addition to places whose markings describe the net's object state, q-nets include a set of cps, the marking of which describes the quantity state. Cps are connected to transitions with undirected arcs – firing a transition connected to one or more cps can change each connected cp's item level. The process model in the centre of Fig. 1 shows a fully defined q-net.

Definition 7 (Quantity Net). *A quantity net is a tripartite graph* $QN = (OCPN, CP, QA)$*, where* $OCPN = (N, pt, F_{var})$ *is an object-centric Petri net with* $N = (P, T, F, l)$*,* CP *is a set of collection points, and* $QA \subseteq (T \times CP)$ *a set of undirected quantity arcs.*

The q-nets defined here differ from those in [6] where the arcs are directed. Despite making the visualisation less expressive, this increases the model's language and removes the strong requirements for their discovery.

For any transition $t \in T$, the set $CP^t \subseteq CP$ describes the cps it is connected to. The overall state of a q-net is a combination of an object state (marking of places) and a quantity state (marking of collection points), i.e., $M = (m_o, m_q) \in M_o \times \mathcal{C}(CP, I)$. An execution binding $b = (t, b_o, b_q)$ of a q-net refers to an object binding and a valid quantity binding $b_q \in \mathcal{C}(CP^t, I)$. A transition is enabled in a marking $M = (m_o, m_q)$ if the underlying $OCPN$ is enabled in m_o. If a binding is executed, the new marking $m^{'} = (m_o^{'}, m_q^{'})$ consists of the new object state $m_o \xrightarrow{(t,\, b_o)} m_o^{'}$ and the new quantity state $m_q^{'} = m_q \oplus b_q$. A q-net can be discovered by identifying the active quantity operations and adding the corresponding cps and arcs to an $OCPN$ mined from the L_{oc}.

Definition 8 (Discovered Q-net). *Let* $QEL = (L_{oc}, I, CP, eqty, \prec_e)$ *be a QEL. The discovered q-net* $QN = (OCPN, CP, QA)$ *is a tuple where* $OCPN$ *is mined from* L_{oc} *using any discovery algorithm,* $CP \subseteq \mathcal{U}_{cp}$ *is the set of cps in the QEL, and* $QA = \{(t, cp) \in T \times CP \mid (a, cp) \in QR \wedge l(t) = a\}$ *the set of quantity arcs with* $QR = \{(act(e), cp) \mid eqty_e^{cp} \neq [\![\,]\!]\}$ *the set of quantity relations.*

The definitions of q-nets and QELs clearly show that the proposed frameworks are extensions to OCELs and OCPNs, thereby not at all limiting their capabilities but merely adding information. Hence, existing methods, techniques and guarantees remain applicable to the QRPM framework, and all behaviour replayable on the OCPN remains replayable on the q-net.

4 Process Intelligence Using QELs

As suggested in [15], we evaluate the feasibility of our framework for the analysis of non-workflow processes by illustrating its capabilities with an example. With

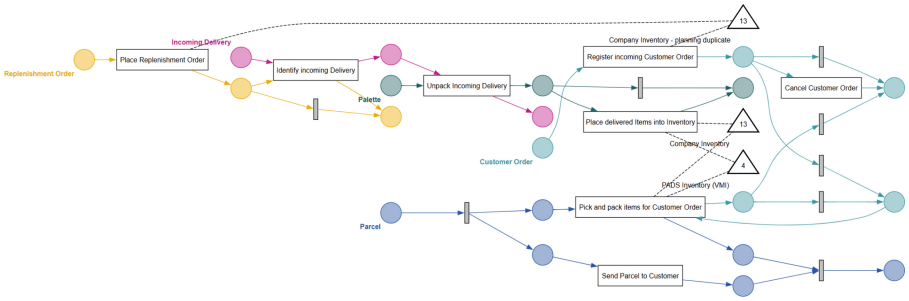

Fig. 4. Q-net discovered from the QEL (only five object types displayed).

a prototypical QEL and implementation of the described framework[1], we demonstrate and discuss its capabilities for the inclusion, detection, and representation of the properties of quantity-related processes.

The Python-based tool *QRPM* (Quantity-related Process Mining) used for this demonstration takes a QEL as an input, which is an OCEL 2.0 in SQLite-format with an additional table *quantity_operations* (like Table 1 without the two columns, "activity" and "timestamp"). Every row refers to an event identifier (*ocel_id*) or the initial item level (*init*), a cp (*ocel_cpid*), as well as one quantity update per item type (columns). The quantity operation table is not required to be complete – active quantity operations are sufficient. QRPM discovers a q-net from an uploaded QEL[2] and detects and visualises the quantity state of every event. It also offers the possibility to perform basic data operations (projections and aggregations) on the quantity operations and quantity state of (selected) events and export this data. For a user-guided analysis, the tool provides descriptive statistics and visualisations on both types of quantity data.

The QEL describing the simulated IMP containing common strategies contains 3622 events of eight activities, with 4097 active quantity operations referring to three collection points and 17 item types. The events are associated with 3441 objects of seven object types; 768 customer orders and 95 replenishment orders. Figure 4 shows the discovered q-net from a sublog only referring to five object types (the two types of employees were removed for easier readability). The numbers in the cps inform of the number of active item types associated with at least one active entry in the quantity operation table. While the underlying OCPN shows two disconnected sub-processes, the discovered q-net can depict a quantity-based connection between them. Furthermore, all quantity operations of the QEL can be represented with the discovered q-net (P1). However, it is unclear what item types the cp refers to and how the execution of an activity impacts its item level in terms of directedness (adding/removing/mixed), involved item types and quantities. This information can be obtained from the QEL and is displayed in QRPM, but it is not part of the model and cannot be incorporated easily, as they may differ among item types. For the discov-

[1] Dataset and implementation: https://github.com/rwth-pads/qrpm.git.
[2] The OCPN is discovered with PM4PY [2] which uses the standard Inductive Miner.

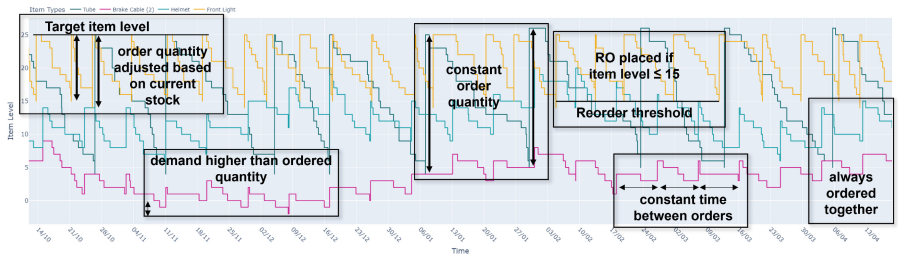

Fig. 5. Quantity State Development of QEL reveals relevant behaviour for the logistics domain.

ery, the frequency of active quantity operations between events of an activity and a cp is not considered – one occurrence in the log is enough for the arc to be added. The current framework is also incapable of detecting or displaying quantity-dependent behaviour (P2 and P3) as it does not consider connections between activities and cps beyond quantity changes, and the quantity state does not restrict the model's behaviour compared to OCPN.

Despite the limitations of the current framework, we now demonstrate the additional process intelligence that can be obtained from the QEL using QRPM and, thereby, emphasise the potential of this framework. On the one hand, including the quantity state allows insights into executed logistics practices such as replenishment policies [3]. Figure 5 shows the planning system's item levels for selected item types. Considering the quantity state, we can see that ROs for tubes and front lights are placed whenever their item level falls below 5 or 15, respectively, indicating a quantity dependency in the execution of the event (P2). We can also spot a quantity dependency in the number of ordered items (P3): While the number of ordered tubes is fixed, the front lights are ordered in varying quantities depending on the item level when the RO is placed, so a target stock level of 25 is achieved. Furthermore, Fig. 5 shows that helmets and brake cables are always ordered and delivered at the same time, independent of their item levels – in fact, the time between orders is constant. As the count of brake cables temporarily sinks below zero, we can deduce that the demand for COs exceeded both the available item level and the reordered quantity. This example shows that when using a QEL, relevant process elements in the logistics domain can now be captured and analysed.

On the other hand, the quantity state offers an additional perspective for determining patterns in the control flow. Figure 5 shows an indication of stock-based triggering of "place RO", which could not be seen without the item levels. The item levels of the vendor-managed inventory and of selected item types of the physical warehouse (Fig. 6) provide further indicators of quantity dependent behaviour. As these cps represent the stock levels of physical warehouses, none of the item levels falls below zero – however, products can be out of stock. Inspecting stock-outs reveals that whenever boxes are out of stock, the stock levels of all products in both warehouses do not decrease until boxes are available again (grey overlay in Fig. 6). Using QRPM to filter for executed events in this

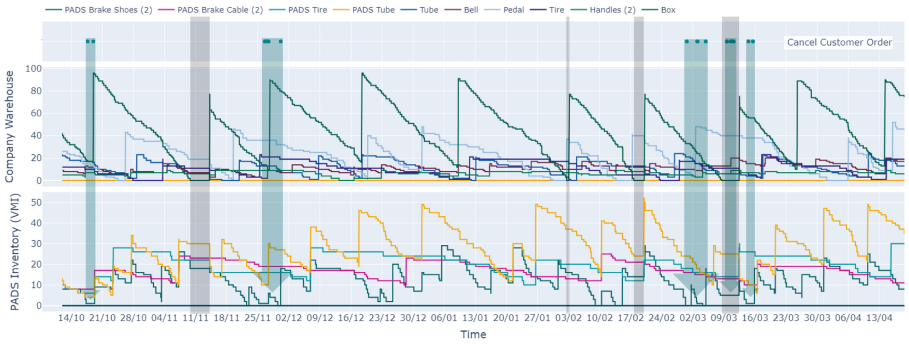

Fig. 6. Dependencies of event executions on stock levels of the physical warehouses. Grey highlighting: periods of box stock-outs, blue-green arrows: cancellations of COs. (Color figure online)

state, we can see that no event of "pick and pack" is among them – this indicates a dependency on the availability of boxes. The top chart in Fig. 6 shows the COs' cancellation, which coincides with stock-outs of the vendor's brake shoes (blue-green arrow). The only deviations occur during the stock-out of boxes, where the availability of five items remains unchanged. An incoming delivery lifts the item level to twelve, which rapidly decreases (by five) to seven. Using QRPM, we can see that the COs for these removals were registered before the cancellations. Finally, we use QRPM to consider the quantity states during the events of "pick and pack" for those COs with multiple executions of this activity (not represented in the Figure). As in the example shown in the introduction, the item level of at least one item type is zero in the company warehouse after every first iteration. In every second iteration, there are only active quantity updates for item types unavailable after the first event.

While standards such as ISA-95 and case studies using inventory data indicate the availability of relational data for creating a QEL, QRPM's practical feasibility remains an open question. Furthermore, we have shown the QEL's capability to convey different kinds of quantity dependency, yet an analyst must still support their detection. The log used in this section is noise-free and perfectly describes the intended behaviour; hence, the presented use case is purely academic and only shows the theoretical possibilities of this framework. However, this work's aim was not the analytical detection of these patterns but to introduce the necessary foundation for doing so. This we have done – by showing that the properties of quantity-related processes can be captured in the proposed QEL and providing a concept, a QEL and an implementation as a basis for further research.

5 Related Work

Several approaches discuss the problems in applying PM techniques to warehouse management processes and introduce methods to overcome this problem.

In all of the proposed methods, the addition of items to a warehouse and the removal of items from a warehouse are integrated into the event log as dedicated activities, thereby allowing for value stream mapping [8], the calculation of additional KPIs [11], or checking the conformance to First-in-First-out principles [4]. The overall state of the involved warehouses is not explicitly considered, and any analysis of the involved material movements requires domain knowledge to identify the relevant activities. The authors of [13] address the need to include additional information into an event log for PM to support the detection of material flows. They advocate the explicit inclusion of locations and introduce the combination of the control flow perspective with the location perspective in separate models. In a previous publication [6], we, rather informally, introduced the idea of including collection points and item types into an event log and suggested a different version of a q-net. In that work, no implementation nor formal definitions and requirements for the detection of the quantity state or the discovery of q-nets were provided.

Processes describing the batched execution of activities or subprocesses are another form of quantity-related process. Several works have addressed the discovery of various types of batched activities [10,12], as well as the detection and modelling of batched sub-processes using BPMN [9]. These works successfully detect this particular type of quantity-dependency without requiring the counting entity and the quantity updates to be provided explicitly. However, they require the batched objects to be uniquely identifiable without allowing for varying quantities or counting types of entities. The authors of [16] address the problem of modelling batched processes, as batched processes exceed the notion of a workflow-like process in which all executions are independent. As in [9] the batched process elements are modelled in a different style to indicate their dependency on other process executions. However, in [16] the process model depicts multiple workflows and shares the batched activities among them to indicate the number of dependent executions and includes data attributes in the semantics of the model.

6 Conclusion

In this work, we introduced a framework for the integration of a quantity state to PM by introducing quantity event logs (QEL) and process models capable of representing counts of items. We have shown the capability of QELs to include typical quantity dependencies of IMPs with a use case and an open-source implementation for a QEL's analysis. While still limited in expressiveness, the proposed extension to OCPNs allows for the representation of an activity's impact on a process' quantity state and the connection of quantity-related subprocesses. Apart from extracting the required data and evaluating the framework's robustness, the detection and modelling of quantity dependencies are areas for future work. Additionally, the consideration of implicit quantity dependencies, as opposed to counts explicitly provided in the log, also poses research questions to be pursued. Overall, this work lays a solid foundation for advancing PM

techniques to better support logistics and inventory management domains by explicitly incorporating quantity-related data.

Acknowledgements. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2023 Internet of Production - 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

1. Berti, A., et al.: OCEL (Object-Centric Event Log) 2.0 Specification (2024)
2. Berti, A., van Zelst, S., Schuster, D.: PM4Py: a process mining library for Python. Softw. Impacts **17**, 100556 (2023)
3. de Kok, T., et al.: A typology and literature review on stochastic multi-echelon inventory models. Eur. J. Oper. Res. **269**(3), 955–983 (2018)
4. Er, M., Astuti, H.M., Wardhani, I.R.K.: Material movement analysis for warehouse business process improvement with process mining. In: Asia Pacific Business Process Management, vol. 219, pp. 115–127. Springer (2015)
5. Garcia, C.D.S., et al.: Process mining techniques and applications – a systematic mapping study. Expert Syst. Appl. **133**, 260–295 (2019)
6. Graves, N., Koren, I., Rafiei, M., van der Aalst, W.M.: From identities to quantities. In: Process Mining Workshops, vol. 503, pp. 462–474. Springer (2024)
7. Jacobi, C., Meier, M., Herborn, L., Furmans, K.: Maturity model for applying process mining in supply chains. Logist. J. (12) (2020)
8. Knoll, D., Reinhart, G., Prüglmeier, M.: Enabling value stream mapping for internal logistics using multidimensional process mining. Expert Syst. Appl. **124**, 130–142 (2019)
9. Martin, N., Pufahl, L., Mannhardt, F.: Detection of batch activities from event logs. Inf. Syst. **95**, 101642 (2021)
10. Martin, N., et al.: Retrieving batch organisation of work insights from event logs. Decis. Support Syst. **100**, 119–128 (2017)
11. Paszkiewicz, Z.: Process mining techniques in conformance testing of inventory processes. In: Business Information Systems Workshops, vol. 160, pp. 302–313. Springer (2013)
12. Pika, A., Ouyang, C., Ter Hofstede, A.H.M.: Configurable Batch-Processing Discovery from Event Logs. ACM (2022)
13. van Cruchten, R.M.E.R., Weigand, H.H.: Process mining in logistics: the need for rule-based data abstraction. In: Proceedings RCIS, pp. 1–9. IEEE (2018)
14. van der Aalst, W.M., Berti, A.: Discovering object-centric petri nets. Fund. Inform. **175**(1–4), 1–40 (2020)
15. van der Aalst, W.M., Reijers, H.A., Maruster, L.: Process mining beyond workflows. Comput. Ind. **161**, 104126 (2024)
16. Wen, Y., Chen, Z., Liu, J., Chen, J.: Mining batch processing workflow models from event logs. Concurr. Comput. Pract. Exp. **25**(13), 1928–1942 (2013)