State-Aware Object-Centric Process Mining: Enhancing OCEL 2.0 with Explicit State Transitions

Dina Kretzschmann[1], Alessandro Berti[1], and Wil M.P. van der Aalst[1,2]

[1] Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
[2] Celonis, Munich, Germany
{dina.kretzschmann, a.berti, wvdaalst}@pads.rwth-aachen.de

Abstract. Modern organizations manage complex processes involving multiple object types, event types, and dynamic attributes, such as stock levels, patient vital signs, or machine status, which define critical object states (e.g., *Understock*, *Patient at Risk*, *Machine Down*). While object-centric process mining (OCPM) with OCEL 2.0 captures these attributes, it does not systematically model state transitions, limiting insights into process dynamics. We propose *State-Aware Object-Centric Process Mining (SA-OCPM)*, an extension of OCEL 2.0 that introduces (1) *object state transition events* to log changes (e.g., *Normal* to *Understock*, *Patient at Risk* to *Stable*, *Machine Down* to *Running*) and (2) *object state-aware events* to refine events with state context (e.g., *Goods Receipt (Understock), Patient Admission (Patient at Risk), Maintenance Start (Machine Down)*). Implemented in a commercial platform, SA-OCPM enables precise analysis of *when*, *why*, and *how* processes deviate from the optimum, as demonstrated in a logistics case study revealing inefficiencies like prolonged understock. SA-OCPM's state-based approach enhances diagnostic granularity and is applicable to domains like healthcare, manufacturing, and customer relationship management.

Keywords: State-Aware Object-Centric Process Mining · OCEL 2.0 · Process Mining · Inventory Management

1 Introduction

Modern organizations manage complex processes involving object types (e.g., materials, patients, machines), event types (e.g., goods receipt, patient admission, maintenance start), and dynamic attributes (e.g., stock levels, vital signs, machine status) that define critical *object states* like *Understock*, *At Risk*, or *Down*. These states impact outcomes, e.g., a material shifting to *Understock* causes order delays, a patient to *Critical* needs urgent care, a machine to *Idle* signals inefficiencies, and a client to *Churned* risks revenue loss. Understanding *when*, *why*, and *how* these transitions occur is key to process optimization. Object-centric process mining (OCPM), based on OCEL 2.0 [2], models object types, event types, and their interactions and attributes, surpassing case-centric

methods. Yet, OCEL 2.0 captures attributes (e.g., stock levels, risk scores) without systematically modeling *object states* or transitions, limiting detection of changes like low stock and links to inefficiencies, thus impeding root cause analysis and insights.

To address this gap, we propose *State-Aware Object-Centric Process Mining (SA-OCPM)*, a structured extension of OCEL 2.0 that explicitly incorporates object states and their transitions. SA-OCPM enhances OCPM by introducing: (1) *object state transition events*, which mark changes like a material shifting from *Normal* to *Understock*, and (2) *object state-aware events*, which refine existing events with state context, such as relabeling *Goods Receipt* as *Goods Receipt (Understock)*. These extensions enable precise tracking of state dynamics, revealing the catalysts and consequences of critical transitions. SA-OCPM is particularly suited for objects with measurable, dynamic attributes that form discrete state sets, such as materials (with stock-based states), patients (with health-based states), or machines (with operational states).

Problem Statement. Existing OCPM approaches, despite modeling objects, events, and attributes, do not explicitly capture *object states* or their transitions, limiting the ability to analyze how state changes drive process behavior. This gap obscures critical insights, such as why a material enters an *Understock* state or how a patient becomes *At Risk*, impeding targeted interventions.

Research Questions. This study addresses the following questions to advance state-aware process analysis:

- RQ1: *State Modeling:* How can object states be defined and integrated into OCEL 2.0 to enable precise tracking of state transitions in object-centric process mining?
- RQ2: *Transition Analysis:* How can state transitions be detected, represented, and analyzed to uncover *when*, *why*, and *how* processes deviate from desirable states?
- RQ3: *Practical Impact:* How does SA-OCPM, applied to domains like inventory management, demonstrate effectiveness and generalizability in identifying and addressing inefficiencies?

Contributions. We introduce SA-OCPM, extending OCEL 2.0 by defining object states (e.g., *Normal*, *Understock*) from dynamic attributes using domain-specific models like Min-Max inventory rules, generating *object state transition events* to log state changes (e.g., *ST CHANGE Normal to Understock*), and refining events into *object state-aware events* (e.g., *Goods Issue (Understock)*) to embed state context. Formalized in Definitions 2 and 3 and implemented in Celonis, SA-OCPM reveals inefficiencies like prolonged *Understock* in a logistics case study, generalizing to domains like healthcare and manufacturing for enhanced process intelligence.

The paper is organized as follows: Section 2 reviews related work. Section 3 details SA-OCPM's framework and formalization. Section 4 presents the implementation. Section 5 presents the case study and results. Section 6 concludes with future research directions.

2 Related Work

In this section, we review literature relevant to our proposed *State-Aware Object-Centric Process Mining (SA-OCPM)*, focusing on its advancements in modeling and analyzing object state transitions within object-centric process mining.

Object-Centric Process Mining: OCPM extends traditional case-centric approaches [1] by modeling multiple object types and their interactions [2]. Techniques include artifact-centric models [18], object-centric behavioral constraint models [17], and object-centric DFGs/Petri nets [3, 6]. Commercial tools like Celonis Process Sphere leverage OCPM for complex process analysis [2]. While OCEL 2.0 captures dynamic attributes (e.g., stock levels), it lacks explicit mechanisms to model state transitions, limiting insights into process dynamics driven by state changes. SA-OCPM addresses this by introducing state transition events and state-aware events, enabling precise tracking of *when*, *why*, and *how* state changes impact processes.

Data-Driven Process Analysis and State-Based Modeling: Data-aware process mining integrates contextual attributes to analyze process behavior. Methods include decision rule discovery [10], multi-perspective process models [20], and simulation-based approaches [19]. State-based models, such as finite state machines [13] and interactive discovery [11], capture dynamic conditions in domains like healthcare [32] and manufacturing [9]. However, these approaches often focus on case-centric states or lack systematic integration of state transitions within object-centric frameworks. SA-OCPM extends OCEL 2.0 by defining states via domain-specific models (e.g., Min-Max inventory rules) and embedding state transition events, enhancing causality analysis across multiple object types.

Inventory Management and Logistics in Process Mining: Process mining has been applied to optimize inventory and logistics [1, 25]. Early case-centric approaches [12, 29] evolved into OCPM to handle multiple object types [2, 23], as seen in case studies [7, 16]. However, OCPM models can become complex due to numerous elements, and prior work [14] introduced activity transformations that complicated analysis. The segmentation framework by [15] addresses this by decomposing object-centric event logs into focused segments based on object relationships, event types, and temporal intervals, simplifying analysis and revealing localized patterns like understock and overstock issues in inventory management. SA-OCPM builds on this by incorporating state-aware events and transition events, enabling focused analysis of state-driven inefficiencies, such as prolonged *Understock* states, in domains like inventory management. Unlike segmentation, which isolates process fragments, SA-OCPM embeds state dynamics directly into the event log, enhancing granularity in tracking *when*, *why*, and *how* state changes impact logistics processes.

Root Cause Analysis and Performance Insights: Root cause analysis in process mining links outcomes to underlying causes using causal modeling [24], machine learning [22], graphical models [30], and domain knowledge [5,25]. These methods are based on models which do not explicitly consider state transitions, limiting their ability to capture dynamic data interactions across objects. SA-

OCPM integrates state transitions directly into the event log, enabling precise identification of catalysts (e.g., supplier delays causing *Understock*) and their process impacts, offering a more granular and actionable approach to root cause analysis.

3 State-Aware Object-Centric Process Mining (SA-OCPM)

We propose *State-Aware Object-Centric Process Mining (SA-OCPM)*, a structured extension of the OCEL 2.0 standard to explicitly incorporate object states and their transitions. Unlike traditional OCPM [2], which models objects, events, and their interactions but does not systematically exploit state dynamics, SA-OCPM enhances OCEL 2.0 by introducing (1) *object state transition events* to mark changes in object states (e.g., from *Normal* to *Understock*) and (2) *object state-aware events* to refine existing events with state-specific labels (e.g., *Goods Receipt (Understock)*). This approach, formalized in Definitions 2 and 3, enables precise analysis of *when*, *why*, and *how* processes evolve due to state

(a) Non-Enriched Object-Centric Event Log

Ev.ID	Mat.	POi	Activity	Supplier	Speed	Stock	Timestamp
E01	M001	PO001	Create Purchase Order Item	Supplier A	Low	75	2025-01-01 09:00
E02	M001	–	Goods Issue	–	–	65	2025-01-02 10:00
E03	M001	–	Goods Issue	–	–	55	2025-01-03 11:00
E04	M001	–	Goods Issue	–	–	45	2025-01-04 12:00
E05	M001	PO001	Goods Receipt	Supplier A	Low	165	2025-01-10 14:00

(b) State-Aware Object-Centric Event Log

Ev.ID	Mat.	POi	Activity	Supplier	Speed	State	Stock	Timestamp
E01	M001	PO001	Create Purchase Order Item (Normal)	Supplier A	Low	Normal	75	2025-01-01 09:00
E02	M001	–	Goods Issue (Normal)	–	–	Normal	65	2025-01-02 10:00
E03	M001	–	Goods Issue (Normal)	–	–	Normal	55	2025-01-03 11:00
E06	M001	–	ST CHANGE Normal to Understock	–	–	Understock	45	2025-01-04 12:00
E04	M001	–	Goods Issue (Understock)	–	–	Understock	45	2025-01-04 12:00
E07	M001	PO001	ST CHANGE Understock to Overstock	Supplier A	Low	Overstock	165	2025-01-10 14:00
E05	M001	PO001	Goods Receipt (Overstock)	Supplier A	Low	Overstock	165	2025-01-10 14:00

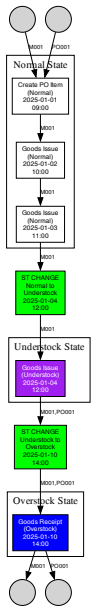

Fig. 1: Comparison of non-enriched (Table 1a) and state-aware (Table 1b) object-centric event logs for inventory management (material with normal stock: 50–150), alongside a visualization of the enriched log (right). The state-aware event log incorporates state data and transition events, detailing *how Understock* resulted from delayed replenishment and *Overstock* from excessive order size, *why* (e.g., Supplier A's delays or low delivery speed), and *when* transitions occurred with precise timestamps.

changes, addressing the limitations of OCPM in capturing dynamic data-driven conditions.

Conceptual Framework: SA-OCPM extends OCEL 2.0, which already models objects, events, and attributes (including data attributes like stock levels or risk scores) as per Definition 1 [2]. While OCEL 2.0 supports dynamic data attributes via the π_{ovmap} function, these are typically used for static or descriptive purposes rather than dynamic state modeling. SA-OCPM introduces a systematic approach to define and track object states, derived from domain-specific data attributes, and integrates them into the event log. This is achieved through two key mechanisms:

1) *Object State Transition Events*: New events are generated to explicitly record when an object's state changes, as defined in Definition 2. For example, a material transitioning from *Normal* to *Understock* due to a stock level dropping below a safety threshold triggers a state change event (e.g., *ST CHANGE Normal to Understock*). These events, timestamped at $t-\epsilon$, link to the affected object and capture the old and new states, enabling traceability of state dynamics.
2) *Object State-Aware Events*: Existing events are refined by appending the object's state at the time of occurrence, as specified in Definition 2. For instance, a *Goods Receipt* event executed when a material is in an *Understock* state is relabeled as *Goods Receipt (Understock)*. This preserves the original event identity while embedding state context, facilitating state-specific process analysis.

Definition 1 (Object-Centric Event Log (OCEL)). *Let $\mathcal{U}\sigma$ denote the universe of all strings. An object-centric event log (OCEL) is a tuple $(E, O, A, OT, \pi_{act}, \pi_{time}, \pi_{ot}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, \leq)$, where E is a finite set of unique events, O is a finite set of unique objects, A is a finite set of activities, OT is a finite set of object types, $\pi_{act} : E \to A$ assigns each event to an activity, $\pi_{time} : E \to \mathbb{R}$ assigns each event a timestamp, $\pi_{ot} : O \to OT$ assigns each object to an object type, $\pi_{omap} : E \to \mathcal{P}(O)$ assigns each event to a set of related objects, $\pi_{vmap} : E \times \mathcal{U}\sigma \nrightarrow \mathcal{U}\sigma \cup \mathbb{R}$ assigns values to event-attribute pairs, $\pi_{ovmap} : O \times \mathcal{U}\sigma \times \mathbb{R} \nrightarrow \mathcal{U}\sigma \cup \mathbb{R}$ assigns values to object-attribute-timestamp triples, and $\leq \subseteq E \times E$ is a total order on events defined by $e_1 \leq e_2 \iff \pi_{time}(e_1) \leq \pi_{time}(e_2)$, with tie-breaking for equal timestamps.*

An OCEL, as in Def. 1, structures process execution data involving multiple interacting entities, with events (E) as process occurrences (e.g., "Order Created"), objects (O) as involved entities (e.g., orders, products), activities (A) as event types linked by π_{act} (e.g., "Create Order"), and object types (OT) categorizing objects via π_{ot} (e.g., "Order"). The function π_{time} assigns timestamps to events, π_{omap} links events to related object sets (e.g., "Ship Package" to order "o1" and package "p1"), π_{vmap} assigns attribute values to event-attribute pairs (e.g., "Channel" as "Online"), π_{ovmap} assigns time-varying attribute values to object-attribute-timestamp triples (e.g., "StockLevel" for "Product X" at time t_1), and $\leq$ orders events temporally by timestamps.

Definition 2 (State-Aware OCEL). *Given an OCEL* $OCEL_{base} = (E, O, A, OT, \pi_{act}, \pi_{time}, \pi_{ot}, \pi_{omap}, \pi_{vmap}, \pi_{ovmap}, \leq_{base})$ *and a specific object attribute name* $\bar{a} \in \mathcal{U}\sigma$ *such that for all* $o \in O$ *and for all* $t \in \mathbb{R}$, $(o, \bar{a}, t) \in \mathrm{dom}(\pi_{ovmap})$ *and* $\pi_{ovmap}(o, \bar{a}, t) \in \mathcal{U}\sigma$ *(This means the attribute* $\bar{a}$ *is always defined for every object* o *at every time* t*, and its value is always a string representing the state of* o *at* t*). A state-aware object-centric event log is a tuple* $OCEL_{state} = (E', O, A, OT, \pi'_{act}, \pi'_{time}, \pi_{ot}, \pi'_{omap}, \pi'_{vmap}, \pi_{ovmap}, \leq')$ *where:*

- $E' = E \cup E''$ *is the extended set of events.*
- $E'' = \{e_{(o,t)} \mid o \in O, t \in \mathbb{R}$ *such that* $\pi_{ovmap}(o, \bar{a}, t) \neq \pi_{ovmap}(o, \bar{a}, t-\epsilon)\}$ *is a set of newly generated* ***state change events****. Here,* ϵ *is an small positive real number representing the time duration just before* t*. (The condition ensures that* $\pi_{ovmap}(o, \bar{a}, t)$ *and* $\pi_{ovmap}(o, \bar{a}, t-\epsilon)$ *are both defined due to the global assumption on* $\bar{a}$*).*
- *The sets* O, A, OT*, and the function* π_{ot} *are inherited from* $OCEL_{base}$*. The function* π_{ovmap} *is also inherited and primarily used for determining states and attribute values.*
- $\pi'_{act} : E' \to A \cup \mathcal{U}\sigma$ *is the activity assignment function, defined as:*
  - $\pi'_{act}(e) = \pi_{act}(e) \oplus$ *"("* $\oplus\, \pi_{ovmap}(o, \bar{a}, t) \oplus$ *")" for* $e \in E$*.*
  - $\pi'_{act}(e_{(o,t)}) =$ *"STCHANGE "* $\oplus \pi_{ovmap}(o, \bar{a}, t-\epsilon) \oplus$ *" to "* $\oplus \pi_{ovmap}(o, \bar{a}, t)$ *for* $e_{(o,t)} \in E''$*, where* $\oplus$ *denotes string concatenation.*
- $\pi'_{time} : E' \to \mathbb{R}$ *is the timestamp assignment function, defined as:*
  - $\pi'_{time}(e) = \pi_{time}(e)$ *for* $e \in E$*.*
  - $\pi'_{time}(e_{(o,t)}) = t - \epsilon$ *for* $e_{(o,t)} \in E''$*.*
- $\pi'_{omap} : E' \to \mathcal{P}(O)$ *is the object mapping function, defined as:*
  - $\pi'_{omap}(e) = \pi_{omap}(e)$ *for* $e \in E$*.*
  - $\pi'_{omap}(e_{(o,t)}) = \{o\}$ *for* $e_{(o,t)} \in E''$*.*
- $\pi'_{vmap} : E' \times \mathcal{U}\sigma \nrightarrow \mathcal{U}\sigma \cup \mathbb{R}$ *is the event attribute value assignment function, defined as:*
  - $\pi'_{vmap}(e, attr) = \pi_{vmap}(e, attr)$ *for* $e \in E$ *and* $(e, attr) \in \mathrm{dom}(\pi_{vmap})$*.*
  - $\pi'_{vmap}(e_{(o,t)}, attr) = \pi_{ovmap}(o, attr, t-\epsilon)$ *for* $e_{(o,t)} \in E''$ *and* $(o, attr, t) \in \mathrm{dom}(\pi_{ovmap})$*. (If* $attr = \bar{a}$*, this gives the new state, which is guaranteed to be defined).*
- $\leq' \subseteq E' \times E'$ *is a total order on the extended set of events, defined as* $e_1 \leq' e_2 \iff \pi'_{time}(e_1) \leq \pi'_{time}(e_2)$*, with a suitable tie-breaking mechanism if needed.*

A State-Aware OCEL, as in Definition 2, extends a standard OCEL by incorporating a state attribute $\bar{a}$ (e.g., "Stock Status"), always defined for every object at all times $t \in \mathbb{R}$, with string values representing object states. It includes original events E and new state change events $E'' = \{e_{(o,t)} \mid o \in O, t \in \mathbb{R}, \pi_{ovmap}(o, \bar{a}, t) \neq \pi_{ovmap}(o, \bar{a}, t-\epsilon)\}$, where ϵ is an small positive duration. For original events, $\pi'_{act}(e) = \pi_{act}(e) \oplus$ "(" $\oplus\, \pi_{ovmap}(o, \bar{a}, t) \oplus$ ")", while for state change events $e_{(o,t)} \in E''$, $\pi'_{act}(e_{(o,t)}) =$ "STCHANGE " $\oplus\, \pi_{ovmap}(o, \bar{a}, t-\epsilon) \oplus$ " to " $\oplus\, \pi_{ovmap}(o, \bar{a}, t)$. Original events retain their timestamps via π'_{time}, while

state change events are timestamped at $t-\epsilon$. The function π'_{omap} preserves original event-object mappings and assigns $\{o\}$ to state change events $e_{(o,t)}$. Event attributes via π'_{vmap} and temporal order $\leq'$ are updated accordingly, making state changes explicit events to facilitate analysis of state transitions and their impact.

Definition 3 (Coalesced State-Aware OCEL). *A Coalesced State-Aware OCEL, denoted $OCEL_{coal}$, is derived from a State-Aware OCEL $OCEL_{state} = (E', O, A, OT, \pi'_{act}, \pi'_{time}, \pi_{ot}, \pi'_{omap}, \pi'_{vmap}, \pi_{ovmap}, \leq')$ (where $E' = E \cup E''$, and E'' is the set of individual state change events) by applying a coalescing transformation. The $OCEL_{coal}$ shares the core components O, A, OT, π_{ot}, and π_{ovmap} with $OCEL_{state}$. The transformation primarily affects the event set and its related mappings as follows:*

- Event Coalescing*:*
  - *Original process events $E \subseteq E'$ from $OCEL_{state}$ are preserved.*
  - *All individual state change events $e'' \in E''$ that share an identical timestamp $t^* = \pi'_{time}(e'')$* and *an identical activity label $\pi'_{act}(e'')$ are grouped. Each such group is replaced by a single new coalesced state change event, $e_c^{(t^*,a)}$, where $a = \pi'_{act}(e'')$ is the common activity label of the grouped events.*
  - *The resulting event set is $E_{coal} = E \cup E''_{coal}$, where E''_{coal} is the set of all such unique coalesced state change events.*
- Properties of New Coalesced Events ($e_c^{(t^*,a)} \in E''_{coal}$) *in $OCEL_{coal}$:*
  - Timestamp*: The event $e_c^{(t^*,a)}$ is assigned the timestamp t^*.*
  - Related Objects*: It is related to the union of all objects that were related to the individual e'' events merged at t^* with activity a, i.e., $\pi''_{omap}(e_c^{(t^*,a)}) = \bigcup\{\pi'_{omap}(e'') \mid e'' \in E'', \pi'_{time}(e'') = t^*, \pi'_{act}(e'') = a\}$.*
  - Activity*: It is assigned the common activity label a, i.e., $\pi''_{act}(e_c^{(t^*,a)}) = a$.*
  - Attributes*: Its attributes can be defined to summarize the merged changes (e.g., a count of affected objects, a list of the specific state transitions involved) or may be sparsely defined/omitted if detailed attribute information for these coalesced events is not critical.*
- Mappings for Original Events*: For events $e \in E$, all their mappings (activity, time, related objects, attributes) remain unchanged from $OCEL_{state}$.*
- Overall Structure*: The new functions $\pi''_{act}, \pi''_{time}, \pi''_{omap}, \pi''_{vmap}$ and the temporal order $\leq''$ for $OCEL_{coal}$ reflect these changes.*

To manage complexity in logs with frequent state changes, SA-OCPM includes a coalescing transformation (Definition 3), which groups simultaneous state change events into single coalesced events. This simplifies analysis by highlighting collective state transitions across multiple objects, as illustrated in Figure 1, where state transitions (e.g., *Normal* to *Understock*) are visualized alongside state-aware events.

Defining Object States: Object states are encoded as an OCEL attribute $\bar{a} \in \mathcal{U}\sigma$, always defined for every object at all times (Definition 2). States (e.g., *Normal*, *Understock*, *Overstock*) are derived during ETL or computed using mathematical models (e.g., optimization or decision rules) based on attribute values. Suitable object types have dynamic, measurable attributes impacting process behavior. Examples include:

- *Materials (Inventory)*: States like *Understock*, *Normal*, *Overstock* from stock levels via Min-Max model [28].
- *Patients (Healthcare)*: States like *Stable*, *At Risk*, *Critical* from clinical metrics (e.g., vital signs).
- *Machines (Manufacturing)*: States like *Running*, *Idle*, *Down* from operational metrics (e.g., uptime).
- *Customers (CRM)*: States like *Active*, *At Risk*, *Churned* from engagement metrics (e.g., purchase frequency).

Objects like orders may aggregate states from related objects (e.g., materials). SA-OCPM focuses on objects with discrete state sets derived from attributes like stock levels or risk scores, ensuring broad applicability.

4 Implementation

This section outlines the implementation of *State-Aware Object-Centric Process Mining (SA-OCPM)* within the Celonis Execution Management System (EMS) to enrich object-centric event logs with state information for granular process analysis.

SA-OCPM implementation begins with loading source data, such as inventory tables from enterprise systems or simulated datasets (e.g., Zenodo artifacts).

Fig. 2: Celonis Dashboard highlighting context-specific metrics (stock structures, material details) and a state-aware object-centric process model.

Object types (e.g., materials (*MAT*), purchase orders (*PO_ITEM*)) and event types are defined in the Celonis Object-Centric Data Model, including their relationships.

State-related information is materialized through custom transformations:

- *Attribute Computation:* Scripts compute dynamic attributes like Economic Order Quantity (EOQ) and Safety Stock (SS) for state definitions, e.g., for material-plant combinations in inventory management.
- *State Transition Events:* SQL scripts detect state changes (e.g., *Normal* to *Understock*) based on attributes, generating *object state transition events* (e.g., *ST CHANGE Normal to Understock*).
- *State-Aware Events:* Transformations postfix activity names with the object's state (e.g., *Goods Receipt* to *Goods Receipt (Understock)*) using query languages like PQL.

These transformations, orchestrated via Celonis Data Jobs, update the process intelligence graph. The enriched event log supports advanced analysis, including state-aware process visualization (e.g., via Multi-Perspective Process Explorer), conformance monitoring (e.g., Min-Max policies), and automated actions (e.g., Action Flows for understock replenishment).

Scripts for state transitions and state-aware events, including SQL and PQL snippets, are available in an OSF dataset (`https://osf.io/7wm2y/?view_only=dd7ca32c15cf4b80884afce82d37357e`). A simulated OCEL and source data are provided on Zenodo (`https://doi.org/10.5281/zenodo.15535073`).

5 Case Study

To evaluate the practical impact of *State-Aware Object-Centric Process Mining (SA-OCPM)*, we applied it to an inventory management scenario at a leading European pet retailer, with 2024 revenues of approximately 4.8 billion €. Conducted in collaboration with the retailer, this study targets inefficiencies tied to object states like *Understock* and *Overstock* across omnichannel operations, which lead to significant financial losses. Leveraging SA-OCPM, we extend prior work [14] by integrating *object state transition events* and *object state-aware events* to analyze state-driven process dynamics, capturing *when*, *why*, and *how* inefficiencies emerge.

5.1 Mathematical Models for Inventory States

To systematically analyze inventory dynamics within SA-OCPM, we define object states, i.e., *Understock*, *Normal*, and *Overstock*, derived from dynamic attributes using the Min-Max method [4, 27]. These states, computed from object-centric event data as outlined in Section 3, enable precise tracking of state transitions, revealing the catalysts and consequences of inefficiencies. Maintaining *Normal* state balances availability and cost, while *Overstock* increases holding costs and *Understock* risks lost sales [8, 21, 31].

The Min-Max method uses four key metrics for each material m:

- *Economic Order Quantity (EOQ)*: Minimizes ordering and holding costs, calculated as $\text{EOQ}_m = \sqrt{\frac{2D_m S}{H}}$, where D_m is annual demand, S is fixed cost per order, and H is holding cost per unit [26].
- *Safety Stock (SS)*: Buffers demand and lead time variability, given by $\text{SS}_m = z \times \sigma_m \times \sqrt{l_m}$, where z is the service level factor, σ_m is demand standard deviation, and l_m is lead time [33].
- *Reorder Point (ROP)*: Triggers replenishment, defined as $\text{ROP}_m = d_m \times l_m + \text{SS}_m$, with d_m as average demand [27].
- *Maximum Stock Level (Max)*: Limits excess inventory, computed as $\text{Max}_m = \text{EOQ}_m + \text{SS}_m$ [27].

Using these, we define the state for material m at time t:

- *Understock*: Inventory $\text{Level}_{m,t} < \text{SS}_m$, signaling insufficient stock.
- *Normal*: $\text{SS}_m \leq$ Inventory $\text{Level}_{m,t} \leq \text{Max}_m$, reflecting optimal levels.
- *Overstock*: Inventory $\text{Level}_{m,t} > \text{Max}_m$, indicating excess.

These states, encoded as attribute $\bar{a}$ in the State-Aware OCEL (Definition 2), enable *object state transition events* (e.g., *ST CHANGE Normal to Understock*) and *object state-aware events* (e.g., *Goods Receipt (Understock)*), linking state changes to process behavior for root cause analysis.

5.2 Object-Centric Event Log

The State-Aware OCEL, constructed via SA-OCPM in Celonis (Section 4), spans December 2022 to April 2024, covering the central warehouse in Germany. It includes 34 event types, enriched with object states (*Understock*, *Normal*, *Overstock*), derived from dynamic attributes per Section 3. The log captures 36,439,490 events across two object types: *materials* (9,648 instances) and *purchase order items* (33,242,938 instances). Relationships are defined as materials linking to zero or more purchase order items (**0..***), and each purchase order item linking to one material (**1..1**), enabling tracking of state dynamics.

Table 1: Overview of event types in the State-Aware OCEL, enriched with object states (*Understock*, *Normal*, *Overstock*) and *object state transition events* per Definition 2, alongside occurrence counts during December 2022 to April 2024.

Goods Issue (Overstock)	24,289,441
Goods Issue (Normal)	8,428,451
Create Sales Order Item (Overstock)	1,684,667
Create Sales Order Item (Normal)	680,844
Goods Issue (Understock)	210,264
Create Purchase Suggestion Item (Overstock)	198,908
Goods Receipt (Overstock)	188,841
Create Purchase Order Item (Overstock)	173,095
Create Purchase Suggestion Item (Normal)	144,586
Create Purchase Order Item (Normal)	121,903
Goods Receipt (Normal)	119,561
Create Sales Order Item (Understock)	93,324
Create Purchase Suggestion Item (Understock)	18,967
Goods Transfer Out (Overstock)	13,822
Create Purchase Order Item (Understock)	9,614
ST CHANGE Normal Overstock	9,396
ST CHANGE Overstock Normal	8,881
ST CHANGE Understock Normal	7,262
ST CHANGE Normal Understock	6,463
Goods Transfer Out (Normal)	5,477
END OVERSTOCK	4,604
END NORMAL	4,093
START OVERSTOCK	3,872
START NORMAL	3,849
Cancel Goods Receipt (Overstock)	3,619
START UNDERSTOCK	1,892
Goods Receipt (Understock)	1,199
Cancel Goods Receipt (Normal)	963
END UNDERSTOCK	916
ST CHANGE Understock Overstock	286
Goods Transfer Out (Understock)	263
Cancel Goods Receipt (Understock)	111
ST CHANGE Overstock Understock	55
Create Purchase Requisition (Normal)	1

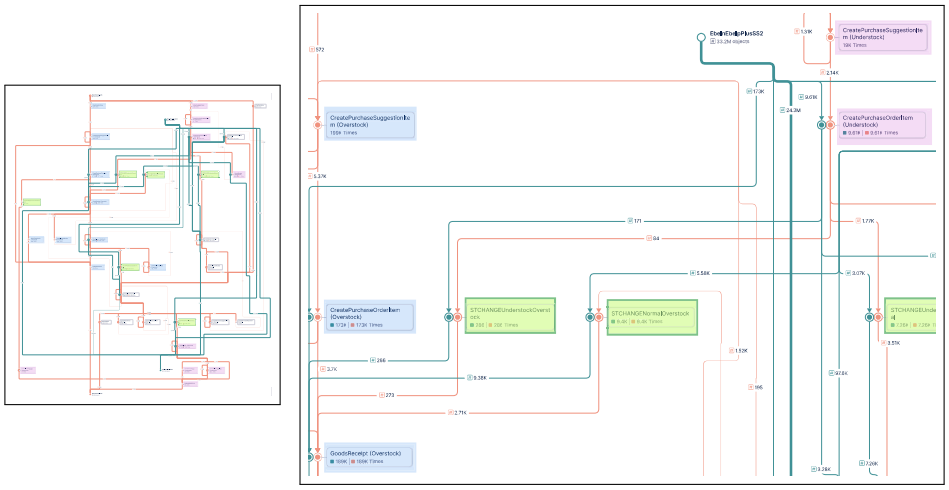

Fig. 3: State-aware object-centric process model derived via SA-OCPM. The left figure (a) displays the full model with Celonis object coloring (e.g., material in orange, purchase order item in dark green), extended to show object states: blue (*Overstock*), purple (*Understock*), white (*Normal*), and light green for *object state transition events* per Definition 2. The right figure (b) zooms in to increase readability.

Object state-aware events (e.g., *Goods Issue (Understock)*) refine activities with state context, while *object state transition events* (e.g., *ST CHANGE Normal to Overstock*) explicitly log state shifts, capturing the moment, prior state, and new state per Definition 2. These link to process activities, ensuring traceability of state-driven dynamics. Table 1 lists event types, associated states, and occurrence counts.

5.3 Process Discovery

Using SA-OCPM, we derived a state-aware, object-centric process model in Celonis with the Multi-Perspective Process Explorer, leveraging the State-Aware OCEL from Section 5.2. This model integrates *object state-aware events* and *object state transition events*, capturing how processes evolve under specific state conditions, per Definition 2.

For clarity, we applied a semantic color-coding scheme in Celonis, as shown in Figure 3. Object types are colored per Celonis defaults (orange for materials, dark green for purchase order items), with manual extensions for states: white for *Normal*, purple for *Understock*, blue for *Overstock*, and light green for *object state transition events*. This visualization highlights state-specific patterns and critical transitions, enhancing diagnostic precision.

The model reveals state-driven behaviors obscured by traditional OCPM. For example, *object state-aware events* like *Goods Receipt (Overstock)* show contributions to prolonged inefficiencies, while *object state transition events* track shifts (e.g., *ST CHANGE Normal to Understock*), supporting targeted analysis of bottlenecks and conformance under specific states, explored further below.

5.4 Main Results

The objective is to identify catalysts of inefficient object states in inventory processes. Using SA-OCPM, we analyze interactions between materials and purchase order items, leveraging *object state-aware events* and *object state transition events* to pinpoint state-driven inefficiencies.

Over the period, 9,648 materials were tracked, with states distributed as 38.5% *Normal*, 37.2% *Overstock*, and 24.3% *Understock* (see Celonis Dashboard, Figure 2). These inefficient states impact inventory worth millions of euros, anonymized for confidentiality. SA-OCPM identifies materials prone to suboptimal states.

The state-aware process model reveals patterns:

- *Create Purchase Order Item (Normal)* transitions to *Goods Receipt (Normal)* in 97,589 cases, but to *Goods Receipt (Overstock)* in 4,473 cases (4.4%) due to excessive quantities, and to *Goods Receipt (Understock)* in 247 cases (0.24%) due to demand underestimation or supplier delays.
- *Create Purchase Order Item (Overstock)* worsens to *Goods Receipt (Overstock)* in 151,478 cases (98%), driven by poor forecasting, with rare shifts to *Goods Receipt (Normal)* (1,124 cases) or *Goods Receipt (Understock)* (7 cases) from unexpected demand or supplier issues.
- *Create Purchase Order Item (Understock)* shifts to *Goods Receipt (Normal)* in 3,276 cases (75%), fails to resolve in 692 cases (16%) to *Goods Receipt (Understock)*, and overcompensates to *Goods Receipt (Overstock)* in 413 cases (9%), signaling reactive ordering.

Analysis of *object state transition events* highlights inefficiencies. In 210,186 cases, materials lingered in *Understock* with active demand but no purchase order, causing 1,460 unfulfilled sales orders. Additionally, 122 materials remained in *Understock* despite multiple *object state-aware events* like *Goods Receipt (Understock)*, due to insufficient quantities. In 1,520 cases, multiple purchase orders during *Overstock* led to excess, reflecting poor synchronization. Recovery times averaged 246 days for *ST CHANGE Overstock to Normal* and 87 days for *ST CHANGE Understock to Normal*, indicating slow corrections.

SA-OCPM's explicit modeling of states and transitions uncovers inefficiencies missed by standard OCPM, such as persistent *Understock* or excessive *Overstock*. By integrating *object state-aware events* and *object state transition events* per Definitions 2 and 3, SA-OCPM pinpoints catalysts like inaccurate forecasting and supply delays, offering actionable insights for optimization across domains with dynamic attributes.

5.5 Process Improvements

To boost inventory efficiency, targeted measures are implemented as automated *Action Flows* in Celonis, triggering interventions based on state-aware, object-centric insights from real-time data. However, full operationalization is pending due to limited integration with systems like SAP.

For example, an Action Flow in Celonis detects understocked materials with active demand but no purchase order. It compiles these into a CSV file, stored

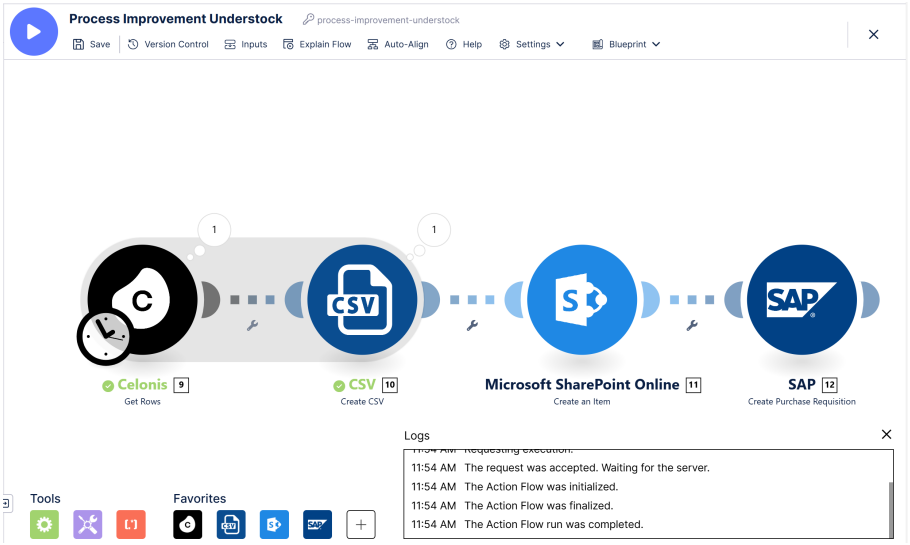

Fig. 4: Celonis Action Flow for Stock Replenishment. Detects understocked materials with demand, generates a CSV list, stores it on SharePoint, and aims to trigger a purchase requisition in SAP. Detection works fully; storage and requisition steps await system integration.

on SharePoint, and aims to create a purchase requisition in SAP. Currently, the final steps remain non-operational due to integration gaps.

Forecasting models now incorporate state awareness to avoid excess purchase orders for overstocked materials and align quantities with demand, reducing stock imbalances. Synchronized purchasing prevents redundant orders, while improved demand forecasting and supplier reliability address understock issues.

Purchasing processes proactively replenish understocked materials with active demand to prevent shortages. Data-driven, state-oriented interventions via Celonis swiftly correct inefficient stock states, boosting inventory responsiveness and supply chain agility.

6 Conclusion and Future Work

This study introduced State-Aware Object-Centric Process Mining (SA-OCPM), a novel extension of OCEL 2.0, to address the critical gap in modeling and analyzing object state transitions within complex processes. Revisiting our research questions, we first tackled **RQ1**. We defined object states (e.g., *Normal*, *Understock*, *Overstock*) derived from dynamic attributes using domain-specific models like the Min-Max method, integrating them into OCEL 2.0 through a state attribute $\bar{a}$ as formalized in Definition 2. This enabled precise tracking of state changes via *object state transition events* (e.g., *ST CHANGE Normal to Understock*). For **RQ2**, SA-OCPM detects and represents state transitions through new events and refines existing ones as *object state-aware events* (e.g., *Goods Receipt (Understock)*), per Definitions 2 and 3. Analysis in Celonis re-

vealed when transitions occur (e.g., timestamps of shifts to *Understock*), why they happen (e.g., supplier delays, poor forecasting), and how they impact processes (e.g., unfulfilled sales orders). Finally, for **RQ3**, our logistics case study at a European pet retailer demonstrated SA-OCPM's effectiveness, uncovering inefficiencies like prolonged *Understock* (e.g., 210,186 cases with active demand but no purchase order) and excessive *Overstock* due to unsynchronized orders. SA-OCPM advances process mining by embedding state dynamics directly into event logs, enhancing diagnostic granularity and enabling state-driven interventions, suggesting generalizability to domains like healthcare and manufacturing with dynamic attributes.

SA-OCPM advances process mining by embedding state dynamics directly into event logs, enhancing diagnostic granularity and enabling state-driven interventions. Future work will focus on automating state detection with machine learning to adapt to complex, non-linear state boundaries, integrating SA-OCPM with real-time systems like SAP for seamless process adjustments, and extending the framework to multi-object state interactions for deeper insights across interconnected entities.

References

1. van der Aalst, W.M.P.: Process mining: data science in action. Springer (2016)
2. van der Aalst, W.M.P.: Object-centric process mining: unraveling the fabric of real processes. Mathematics **11**(12), 1–11 (2023)
3. van der Aalst, W.M.P., Berti, A.: Discovering object-centric Petri nets. Fundam. Informaticae **175**(1-4), 1–40 (2020)
4. Asana, I.M.D.P., Radhitya, M.L., Widiartha, K.K., Santika, P.P.: Inventory control using ABC and min-max analysis on retail management information system. In: Proc. J. Phys.: Conf. Ser. vol. 1469, p. 012097. IOP Publishing (2020)
5. Badakhshan, P., Wurm, B., Grisold, T., Geyer-Klingeberg, J.: Creating business value with process mining. J. Strateg. Inf. Syst. **31**(4), 101745 (2022)
6. 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)
7. Berti, A., Jessen, U., Park, G., Rafiei, M.: Analyzing interconnected processes: using object-centric process mining to analyze procurement processes. Int. J. Data Sci. Anal. (2023)
8. Brabänder, C.: Modell 4: Kontinuierliches Bestandsmanagement. Stochastisches Bestandsmanagement pp. 75–92 (2020)
9. Celik, U., Yurtay, Y.: Improvement of assemble-to-order model processes with process mining: dynamic analysis and hybrid approaches. IEEE Access (2025)
10. De Leoni, M., van der Aalst, W.M.P.: Data-aware process mining: discovering decisions in processes using alignments. In: Proc. ACM Symp. Appl. Comput. pp. 1454–1461 (2013)
11. van Eck, M.L., Sidorova, N., van der Aalst, W.M.P.: Discovering and exploring state-based models for multi-perspective processes. In: BPM. pp. 142–157. Springer (2016)
12. ER, M., Astuti, H.M., Wardhani, I.R.K.: Material movement analysis for warehouse business process improvement with process mining: a case study. In: AP-BPM. pp. 115–127. Springer (2015)

13. Hompes, B.F.A., van der Aalst, W.M.P.: Lifecycle-based process performance analysis. In: CoopIS, CTC, ODBASE. pp. 336–353. Springer (2018)
14. Kretzschmann, D., Berti, A., van der Aalst, W.M.P.: A data-driven framework for retail inventory optimization: integrating object-centric process mining and mathematical models. (2025)
15. Kretzschmann, D., Berti, A., van der Aalst, W.M.P.: Segmentation for Optimizing Long-Lifecycle Processes in Object-Centric Process Mining. (2025)
16. Kretzschmann, D., Park, G., Berti, A., van der Aalst, W.M.P.: Overstock problems in a purchase-to-pay process: an object-centric process mining case study. In: CAiSE Workshops. pp. 347–359. Springer (2024)
17. Li, G., de Carvalho, R.M., van der Aalst, W.M.P.: Automatic discovery of object-centric behavioral constraint models. In: BIS. pp. 43–58. Springer (2017)
18. Lu, X., Nagelkerke, M., Van De Wiel, D., Fahland, D.: Discovering interacting artifacts from ERP systems. IEEE Trans. Serv. Comput. **8**(6), 861–873 (2015)
19. López-Pintado, O., Murashko, S., Dumas, M.: Discovery and simulation of data-aware business processes. In: Proc. Int. Conf. Process Min. (ICPM). pp. 105–112. IEEE (2024)
20. Mannhardt, F.: Multi-perspective process mining. Ph.D. thesis, Eindhoven Univ. Technol. (2018)
21. Meyer, J.C., Sander, U., Wetzchewald, P.: Bestände senken, Lieferservice steigern: Ansatzpunkt Bestandsmanagement. FIR e. V. an der RWTH Aachen (2019)
22. Nadim, K., Ragab, A., Ouali, M.S.: Data-driven dynamic causality analysis of industrial systems using interpretable machine learning and process mining. J. Intell. Manuf. **34**(1), 57–83 (2023)
23. Oldenburg, F., Hoberg, K., Leopold, H.: Process mining in supply chain management: state-of-the-art, use cases and research outlook. Int. J. Prod. Res. pp. 1–16 (2024)
24. Qafari, M.S., van der Aalst, W.: Root cause analysis in process mining using structural equation models. In: BPM Workshops. pp. 155–167. Springer (2020)
25. Reinkemeyer, L.: Process mining in action. Process Min. Action Princ. Use Cases Outlook (2020)
26. Riza, M., Purba, H.H., Mukhlisin: The implementation of economic order quantity for reducing inventory cost. Res. Logist. Prod. **8**(3), 207–216 (2018)
27. Rizqi, Z.U., Khairunisa, A.: Integration of deterministic and probabilistic inventory methods to optimize the balance between overstock and stockout. In: Proc. IOP Conf. Ser.: Mater. Sci. Eng. vol. 722, p. 012060. IOP Publishing (2020)
28. Scarf, H.E., Arrow, K.J., Karlin, S.: A min-max solution of an inventory problem. Rand (1957)
29. Schuh, G., Gutzlaff, A., Cremer, S., Schoppen, M.: Understanding process mining for data-driven optimization of order processing. Procedia Manuf. **45**, 417–422 (2020)
30. Tang, J., Liu, Y., Lin, K.Y., Li, L.: Process bottlenecks identification and its root cause analysis using fusion-based clustering and knowledge graph. Adv. Eng. Inform. **55**, 101862 (2023)
31. Tempelmeier, H.: Bestandsmanagement in supply chains. BoD–Books on Demand (2005)
32. Valero-Ramon, Z., Fernandez-Llatas, C., Valdivieso, B., Traver, V.: Dynamic models supporting personalised chronic disease management through healthcare sensors with interactive process mining. Sensors **20**(18), 5330 (2020)
33. Waters, D.: Inventory control and management. Wiley (2003)