Complex Systems Informatics and Modeling Quarterly (CSIMQ)
eISSN: 2255-9922
Published online by RTU Press, https://csimq-journals.rtu.lv
Article 264, Issue 47, June/July 2026, Pages 84–104
https://doi.org/10.7250/csimq.2026-47.04

Managing Complexity in Object-Centric Process Mining: A Cohesion-Driven Decomposition and Aggregation Framework

Khalil Mecheraoui[1,2*]

[1] ICOSI Laboratory, Abbes Laghrour University, Khenchela, 40004, Algeria
[2] Higher National School of Forests, Khenchela, 40000, Algeria

mecheraoui.khalil@ensf.dz

Abstract. Object-Centric Process Mining (OCPM) often discovers models that are too dense and complex for effective human comprehension. To address this challenge, this work proposes a novel framework for the automated decomposition and aggregation of Object-Centric Petri Nets (OCPNs). The framework is based on the principles of high cohesion and low coupling. First, the Composite Cohesion Metric is introduced to measure the relational strength between object types from three distinct perspectives: behavioral synchronization, structural composition, and empirical log-based co-occurrence. By mapping these measures to a weighted graph and applying modularity-based community detection, the overall model is partitioned into semantically coherent sub-process modules. Finally, an Aggregated Net is constructed to offer a high-level architectural view of the system. The proposed framework is evaluated using the standard Order Management benchmark. The results demonstrate that the approach used successfully structures the process into distinct logical execution layers. The approach correctly assigns resource objects to their functional modules. The proposed framework significantly reduces the complexity of the model without affecting its global validity.
Keywords: Object-Centric Process Mining, Object-Centric Petri Nets, Process Decomposition, Process Architecture, Cohesion and Coupling.

1 Introduction

Process mining is a discipline that helps organizations understand and optimize real-world processes using event data. It provides a set of techniques to discover descriptive models, check the conformance of prescriptive models, and improve processes based on the event data they generate [1], [2].

Traditional process mining techniques assume that each event in a log corresponds to exactly one process instance (a unique case identifier). This case-centric view is insufficient for enterprise systems (e.g., ERP and SCM). In these environments, business processes involve multiple

* Corresponding author

© 2026 Khalil Mecheraoui. This is an open-access article licensed under the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0).

Reference: K. Mecheraoui, "Managing Complexity in Object-Centric Process Mining: A Cohesion-Driven Decomposition and Aggregation Framework," Complex Systems Informatics and Modeling Quarterly, CSIMQ, no. 47, pp. 84–104, 2026. Available: https://doi.org/10.7250/csimq.2026-47.04

Additional information. Author's ORCID iD: K. Mecheraoui – https://orcid.org/0000-0001-9906-6074. PII S225599222600264X. Received: 28 February 2026. Accepted: 28 July 2026. Available online: 31 July 2026.

interacting objects of different types – such as ORDERS, ITEMS, PACKAGES, and CUSTOMERS – with complex one-to-many and many-to-many relationships. Forcing these interactions into a single case notion often leads to *convergence* (where one case is copied across many objects) and *divergence* (where many different object behaviors are flattened into a single trace). These issues lead to incorrect analysis [3].

Object-Centric Process Mining (OCPM) [3], [4] addresses this limitation. It enables the discovery of expressive process models, such as Object-Centric Petri Nets (OCPNs) [5], from Object-Centric Event Logs (OCELs) [6]. However, the enhanced expressiveness of OCPNs comes at the cost of complexity. Models discovered from real-world systems often contain many interacting object types with high connectivity. These models frequently become too complex and dense for effective human comprehension. Furthermore, a straightforward decomposition approach, such as creating a single module per object type, often severs logical connections between objects that inherently belong together. For instance, in an e-commerce system, separating ORDERS from ITEMS would destroy the semantic integrity of the process. Therefore, a simplification strategy that respects the underlying architectural semantics of the system is required.

This article presents a novel framework for an efficient, object-aware decomposition and aggregation of OCPNs. The software engineering principles of high cohesion and low coupling [7] and modular information hiding [8] guide this framework. The approach used groups object types that are strongly related structurally, behaviorally, and empirically into cohesive sub-process modules. Figure 1 illustrates the proposed framework. It takes the overall model and log as input and constructs a hierarchical business process architecture. The process consists of three phases: Cohesion Quantification, Structural Decomposition, and Model Aggregation.

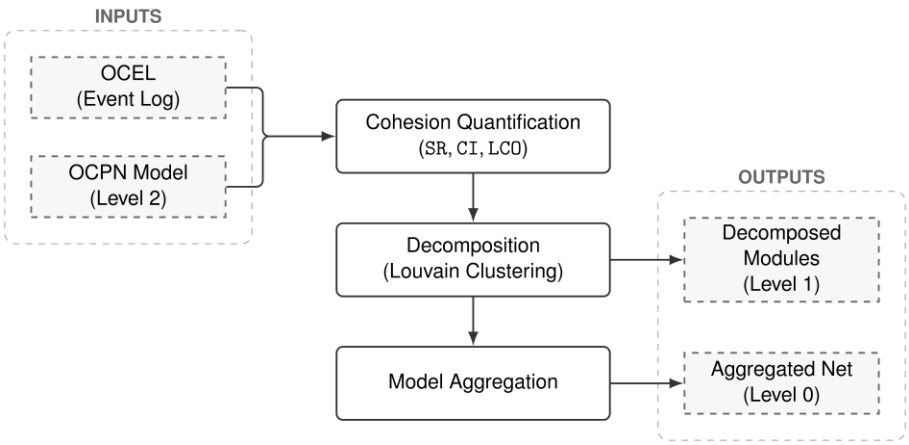

Figure 1. Overview of the proposed framework

The research method follows the following main activities:

1. First, introduce the Composite Cohesion Metric that quantifies the relational strength between object types. To obtain a holistic measure, combine three metrics covering distinct perspectives: the behavioral *Synchronization Ratio* (SR), the structural *Composition Index* (CI), and the empirical *Log-Based Co-occurrence* (LCO). The data-driven perspective is pivotal for dealing with the complexity of processes. Real execution records often reveal that structurally similar connections differ in actual significance. For instance, two types of objects may have high (SR) and (CI) scores even if their interaction rarely occurs in reality [3].
2. Next, use the Louvain community detection method [9] to partition the complex OCPNs into meaningful sub-process modules. Within each module, the internal logic is visible, whereas the external synchronization is encapsulated within interface transitions. This modularity-based method is chosen since it aligns with the principles of cohesion and coupling. It does not require prior knowledge regarding the number of clusters.

3. Finally, to obtain a holistic view of the system, introduce the *Aggregated Net*. This high-level model is derived directly from the decomposed modules and their interface transitions. The analyst looks at the Aggregated Net to understand the global topology and dependencies between interacting modules.

In general, a three-level hierarchical analysis framework is proposed that includes: the Aggregated Net (level 0) as a high-level architectural view, the Decomposed System (level 1) as a modular execution view, and the Original OCPN (level 2), which remains the mathematical foundation for execution validity and global conformance checking.

The remainder of this article is organized as follows. Section 2 situates the work within the state-of-the-art. Section 3 provides the basic definitions of object-centric event logs and Petri nets. Section 4 introduces an e-commerce scenario that serves as the running example. Section 5 constitutes the core of the article. It details the decomposition and aggregation framework. Section 6 is devoted to experimental results and discussion. Finally, Section 7 concludes the article and suggests future research perspectives.

2 Related Work

The complexity of discovering and checking conformance for large and complex process models and event logs has long been a challenge in process mining. To address this, divide-and-conquer strategies have been extensively studied [10]–[20].

2.1 Complexity Management in Traditional Process Mining

An interesting decomposition framework is presented in [11]. The author proposed a so-called valid decomposition of the process model to enable the decomposition of process discovery and conformance checking problems. He showed that such problems can be split into smaller sub-problems that can be analyzed easily, and their results combined into solutions for the original problems. This foundational work was further refined and complemented in the literature [12]–[14]. Similarly, traditional trace clustering techniques [16] attempt to reduce complexity by grouping similar process instances. However, these strategies are often designed for flattened event logs or result in artificial partitions. That is, the decomposed nets do not represent a logical functional division of the system.

In [15], a natural compositional conformance checking approach for Multi-Agent Systems (MAS) is proposed. Nested Petri Nets (NP-nets) [21] are used, in which tokens in a system net represent agents with their own internal workflow nets. They formally demonstrated that the fitness of the global multi-agent system can be verified by projecting the overall event log onto the system net and individual agent sub-nets and checking each projection in isolation.

Although the aforementioned methods effectively reduce computational load, they are designed for flattened case-centric models. They do not account for the complexity of multiple interacting object lifecycles. An overview of the challenges that arise when object lifecycles of different types interact in one-to-many or many-to-many relationships is detailed in [22].

2.2 Complexity Management in Object-Centric Process Mining

Object-Centric Process Mining (OCPM) allows the discovery of expressive process models, such as Object-Centric Petri Nets (OCPNs) [5], from Object-Centric Event Logs (OCELs) [6]. However, the enhanced expressiveness of object-centric models comes at the cost of complexity due to the high connectivity between object types.

In [17], the authors demonstrated the significance of multi-level information in hierarchical process mining. They showed how abstraction hierarchies can simplify analysis. This research

extends the concept to the object-centric domain, offering a three-level hierarchical analysis framework that bridges architectural abstraction and executable semantics.

Recent contributions have specifically addressed the challenge of managing complexity through object type analysis. In [18], Markov Directly-Follow Multigraphs are introduced to cluster object types. This method does not take into account structural one-to-many dependencies or empirical co-occurrence frequencies—factors that are central to the composite cohesion metric.

In a parallel direction, multi-dimensional data operations—like drill-down and roll-up—are proposed for OCPM [19], [20]. These methods focus on the operational granularity at the log level. They do not deal directly with the decomposition of OCPNs, which are the standard Petri net extension used in the field. The proposed framework fills this gap by establishing an architectural modularization of OCPNs guided by the software engineering principles of cohesion and coupling.

In summary, existing methods only partially solve the problem of OCPM complexity. None of them fully combine behavioral synchronization, structural composition (specifically variable arcs), and empirical log-based evidence into a cohesive software engineering framework. The research discussed in this article fills this gap by providing a three-level hierarchical decomposition of OCPNs.

3 Preliminaries

This section recalls the foundational definitions required for the proposed framework. Let D be a set. Denote by $\mathcal{P}(D)$ the power set of D. Assume the existence of pairwise disjoint universes for objects $\mathcal{U}_{\text{obj}}$, object types $\mathcal{U}_{\text{ot}}$, activities $\mathcal{U}_{\text{act}}$, and events $\mathcal{U}_{\text{ev}}$.

Real-world processes often involve multiple interacting objects (e.g., an order, a customer, and a package) that cannot be flattened into a single case notion without losing data. In object-centric process mining, a single event may be associated with multiple objects of different types.

Definition 1 (Object-Centric Event Log). *An* object-centric event log *is a tuple* $L = (E, O, \texttt{oType}, \texttt{Act}, \texttt{Obj})$*, where:*

- $E \subseteq \mathcal{U}_{\text{ev}}$ *is a finite set of events;*
- $O \subseteq \mathcal{U}_{\text{obj}}$ *is a finite set of objects;*
- $\texttt{oType}: O \to \mathcal{U}_{\text{ot}}$ *is a function mapping each object to its type;*
- $\texttt{Act}: E \to \mathcal{U}_{\text{act}}$ *is a function mapping each event to an activity;*
- $\texttt{Obj}: E \to \mathcal{P}(O)$ *is a function mapping each event to a (non-empty) set of involved objects.*

Labeled Petri Nets (LPNs) are used as the underlying process modeling formalism. In this formalism, transitions are labeled with observable activities or the silent label τ.

Definition 2 (Labeled Petri Net). *A* labeled Petri net *is a tuple* $N = (P, T, F, l)$*, where:*

- P *is a finite set of places;*
- T *is a finite set of transitions such that* $P \cap T = \emptyset$*;*
- $F \subseteq (P \times T) \cup (T \times P)$ *is the flow relation (set of arcs);*
- $l: T \to \mathcal{U}_{\text{act}} \cup \{\tau\}$ *is a labeling function mapping transitions to visible activities or the silent label* τ*.*

Object-Centric Petri Nets (OCPNs) extend labeled Petri nets by explicitly associating places with object types. This extension enables the joint modeling of multiple interacting object lifecycles.

A defining characteristic of the OCPN formalism is the *variable arc*. Unlike ordinary arcs, variable arcs allow a transition to consume and produce a variable number of tokens in a single firing. This mechanism allows OCPNs to capture one-to-many relationships.

Definition 3 (Object-Centric Petri Net). *An* object-centric Petri net *is a tuple* $ON = (N, \texttt{pType}, \mathcal{V}_{\text{arc}})$*, where:*

- $N = (P, T, F, l)$ *is a labeled Petri net;*
- $\mathrm{pType}: P \to \mathcal{U}_{\mathrm{ot}}$ *is a function mapping each place to a single object type;*
- $\mathcal{V}_{\mathrm{arc}} \subseteq F$ *is the subset of variable arcs.*

The formal execution semantics (markings and firing rules) are not discussed here as the approach used analyzes OCPNs exclusively from a structural perspective complemented by empirical evidence from event data.

4 Running Example: An E-commerce Process

The approach is illustrated using a simplified e-commerce scenario that will serve as a running example throughout the article. The example involves multiple interacting object types, namely: CUSTOMER, ITEM, MERCHANT, and COUPON. It exhibits both one-to-one and one-to-many relationships. This makes it representative of real-world object-centric processes. Section 4.1 presents an example with an object-centric process model and Section 4.2 shows the corresponding event data. The challenge is briefly discussed in Section 4.3.

4.1 Object-Centric Process Model

Figure 2 illustrates an OCPN $\mathcal{N}$. It models the lifecycle of four object types: CUSTOMER, ITEM, MERCHANT, and COUPON. Places in $\mathcal{N}$ are typed by object types, while transitions represent business activities.

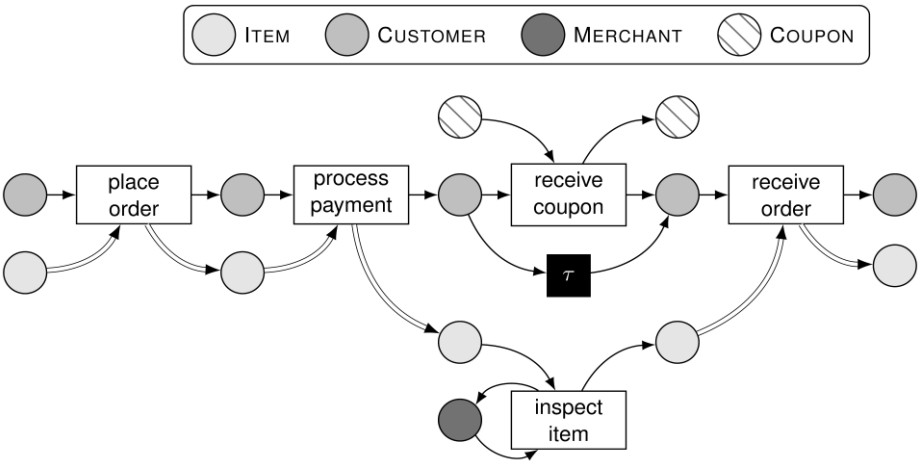

Figure 2. An OCPN $\mathcal{N}$ for an e-commerce process with ITEM (light gray), CUSTOMER (gray), MERCHANT (dark gray), and COUPON (hatched) object types

The process begins when a customer places an order for a set of items. This is modeled by the transition 'place order', which consumes a single CUSTOMER token and an arbitrary number of ITEM tokens. Such behavior captures a one-to-many relationship between customers and items. The subsequent activity 'process payment' synchronizes the same object types.

After the payment is processed, the lifecycle splits. On the one hand, items are sent for inspection by a merchant. This is modeled by the transition 'inspect item' which synchronizes a single item with a single merchant. On the other hand, the customer may optionally receive a coupon through the transition 'receive coupon'. This activity involves the CUSTOMER and COUPON object types and is not related to items. A silent transition is used to model optional behavior without introducing additional activities.

Finally, once all items are inspected, the customer receives the order along with all the corresponding items via the 'receive order' transition.

This OCPN $\mathcal{N}$ illustrates (1) variable arcs that bind CUSTOMER to ITEM in a one-to-many relationship, (2) the divergence of paths between ITEM and CUSTOMER during inspection, and (3) the silent transition (τ) making the coupon interaction optional.

4.2 Object-Centric Event Log

The OCEL fragment *L* in Table 1 illustrates an execution of the e-commerce system modeled in Figure 2.

Table 1. Fragment of an Object-Centric Event Log (OCEL). Columns correspond to CUSTOMER, ITEM, MERCHANT, and COUPON object types.

Event ID	Activity	CUSTOMER	ITEM	MERCHANT	COUPON
e_{01}	place order	cst_1	$\{\mathrm{i}_1, \mathrm{i}_2\}$	–	–
e_{02}	process payment	cst_1	$\{\mathrm{i}_1, \mathrm{i}_2\}$	–	–
e_{03}	place order	cst_2	$\{\mathrm{i}_3\}$	–	–
e_{04}	process payment	cst_2	$\{\mathrm{i}_3\}$	–	–
e_{05}	place order	cst_3	$\{\mathrm{i}_4, \mathrm{i}_5, \mathrm{i}_6\}$	–	–
e_{06}	process payment	cst_3	$\{\mathrm{i}_4, \mathrm{i}_5, \mathrm{i}_6\}$	–	–
e_{07}	inspect item	–	$\{\mathrm{i}_1\}$	mch_1	–
e_{08}	inspect item	–	$\{\mathrm{i}_3\}$	mch_2	–
e_{09}	receive order	cst_2	$\{\mathrm{i}_3\}$	–	–
e_{10}	inspect item	–	$\{\mathrm{i}_2\}$	mch_1	–
e_{11}	receive order	cst_1	$\{\mathrm{i}_1, \mathrm{i}_2\}$	–	–
e_{12}	inspect item	–	$\{\mathrm{i}_4\}$	mch_2	–
e_{13}	inspect item	–	$\{\mathrm{i}_5\}$	mch_2	–
e_{14}	inspect item	–	$\{\mathrm{i}_6\}$	mch_2	–
e_{15}	place order	cst_4	$\{\mathrm{i}_7\}$	–	–
e_{16}	process payment	cst_4	$\{\mathrm{i}_7\}$	–	–
e_{17}	place order	cst_5	$\{\mathrm{i}_8, \mathrm{i}_9\}$	–	–
e_{18}	process payment	cst_5	$\{\mathrm{i}_8, \mathrm{i}_9\}$	–	–
e_{19}	receive order	cst_3	$\{\mathrm{i}_4, \mathrm{i}_5, \mathrm{i}_6\}$	–	–
e_{20}	place order	cst_6	$\{\mathrm{i}_{10}, \mathrm{i}_{11}, \mathrm{i}_{12}, \mathrm{i}_{13}\}$	–	–
e_{21}	process payment	cst_6	$\{\mathrm{i}_{10}, \mathrm{i}_{11}, \mathrm{i}_{12}, \mathrm{i}_{13}\}$	–	–
e_{22}	inspect item	–	$\{\mathrm{i}_7\}$	mch_1	–
e_{23}	inspect item	–	$\{\mathrm{i}_8\}$	mch_2	–
e_{24}	inspect item	–	$\{\mathrm{i}_9\}$	mch_2	–
e_{25}	inspect item	–	$\{\mathrm{i}_{10}\}$	mch_2	–
e_{26}	inspect item	–	$\{\mathrm{i}_{11}\}$	mch_2	–
e_{27}	inspect item	–	$\{\mathrm{i}_{12}\}$	mch_2	–
e_{28}	receive order	cst_4	$\{\mathrm{i}_7\}$	–	–
e_{29}	receive order	cst_5	$\{\mathrm{i}_8, \mathrm{i}_9\}$	–	–
e_{30}	inspect item	–	$\{\mathrm{i}_{13}\}$	mch_2	–
e_{31}	place order	cst_7	$\{\mathrm{i}_{14}\}$	–	–
e_{32}	process payment	cst_7	$\{\mathrm{i}_{14}\}$	–	–
e_{33}	place order	cst_8	$\{\mathrm{i}_{15}, \mathrm{i}_{16}, \mathrm{i}_{17}\}$	–	–
e_{34}	process payment	cst_8	$\{\mathrm{i}_{15}, \mathrm{i}_{16}, \mathrm{i}_{17}\}$	–	–
e_{35}	receive coupon	cst_8	–	–	cp_1
e_{36}	inspect item	–	$\{\mathrm{i}_{14}\}$	mch_1	–
e_{37}	inspect item	–	$\{\mathrm{i}_{15}\}$	mch_1	–
e_{38}	inspect item	–	$\{\mathrm{i}_{16}\}$	mch_1	–
e_{39}	inspect item	–	$\{\mathrm{i}_{17}\}$	mch_1	–
e_{40}	receive order	cst_8	$\{\mathrm{i}_{15}, \mathrm{i}_{16}, \mathrm{i}_{17}\}$	–	–
e_{41}	receive order	cst_7	$\{\mathrm{i}_{14}\}$	–	–
e_{42}	receive order	cst_6	$\{\mathrm{i}_{10}, \mathrm{i}_{11}, \mathrm{i}_{12}, \mathrm{i}_{13}\}$	–	–

The fragment captures the lifecycle of the four object types – CUSTOMER, ITEM, MERCHANT, and COUPON – with object instances $\{\mathsf{cst}_1, \ldots, \mathsf{cst}_8\}$, $\{\mathsf{i}_1, \ldots, \mathsf{i}_{17}\}$, $\{\mathsf{mch}_1, \mathsf{mch}_2\}$, and $\{\mathsf{cp}_1\}$, respectively. Each event in the log records an executed activity together with the set of involved object instances. These records involve varying numbers of items. For instance, event e_{01} records customer cst_1 placing an order for two items, whereas event e_{20} records an order for four items.

Inspection events (e.g., e_{07} and e_{10}) show that items are handled individually by merchants. COUPON-related behavior appears only once in the log, where customer cst_8 receives a coupon in event e_{35}. Although this interaction is present in the process model, its infrequent occurrence in the log indicates weak empirical coupling between CUSTOMER and COUPON.

4.3 Challenge

This scenario illustrates that even a simplified real-world process involves complex interactions between multiple object types. Furthermore, it demonstrates that analyzing such systems through a single perspective is insufficient to capture the full process reality. For instance, considering only the model structure risks obscuring the frequency of execution. Hence, overstating the significance of rare interactions such as receiving a coupon.

This observation leads to the central research question: *How can diverse relational strengths be quantified and integrated to automatically decompose the model into simpler, semantically coherent, and manageable sub-processes?*

5 Decomposition and Aggregation of OCPNs

This section proposes a framework to decompose complex OCPNs into semantically coherent sub-processes and derive an aggregated architectural view.

The methodology proceeds, as discussed in the Introduction, in four steps. First, quantify the relational strength between object types using a multi-perspective cohesion metric. Second, map these measurements to a weighted graph to identify semantic clusters using community detection. Third, utilize these clusters as a blueprint to decompose the original OCPN into sub-process modules. Fourth, to obtain a holistic view of the system, introduce the Aggregated Net. This high-level architectural model, formalized as a classical Petri net, is derived directly from the decomposed modules and their interface transitions.

5.1 Measuring Composite Cohesion

Let $ON = ((P, T, F, l), \mathtt{pType}, \mathcal{V}_{\mathrm{arc}})$ be an OCPN and let $L = (E, O, \mathtt{oType}, \mathtt{Act}, \mathtt{Obj})$ be an OCEL. To measure the relational strength between object types, the necessary sets are first defined. $\mathrm{E}(ot)$ is defined as the set of events associated with an object type $ot \in \mathcal{U}_{\mathrm{ot}}$:

$$\mathrm{E}(ot) = \{e \in E \mid \exists o \in \mathtt{Obj}(e) \colon \mathtt{oType}(o) = ot\}. \tag{1}$$

$\mathrm{E}(ot)$ represents the specific subset of the log relevant to the lifecycle of object type ot. In addition, $\mathtt{tType}(t)$ is the set of object types connected to transition t:

$$\mathtt{tType}(t) = \{\mathtt{pType}(p) \mid p \in P \wedge ((p,t) \in F \vee (t,p) \in F)\} \tag{2}$$

and $\mathrm{T}(ot)$ is the set of transitions associated with a specific object type $ot \in \mathcal{U}_{\mathrm{ot}}$:

$$\mathrm{T}(ot) = \{t \in T \mid ot \in \mathtt{tType}(t)\}. \tag{3}$$

For instance, consider the OCPN $\mathcal{N}$ in Figure 2 and the log L in Table 1. For the transition 'place order' and the object type ITEM:

$$\begin{gathered}\mathtt{tType}(\text{place order}) = \{\textsc{Customer}, \textsc{Item}\} \\ \mathtt{T}(\textsc{Item}) = \{\text{place order, process payment, inspect item, receive order}\}\end{gathered} \tag{4}$$

Furthermore, considering the object types of CUSTOMER and COUPON in the event log L, the sets of associated events are:

$$\begin{gathered}\mathrm{E}(\textsc{Customer}) = \{e_{01}, \dots, e_{06}, e_{09}, e_{11}, e_{15}, \dots, e_{21}, e_{28}, e_{29}, e_{31}, \dots, e_{35}, e_{40}, \dots, e_{42}\} \\ \mathrm{E}(\textsc{Coupon}) = \{e_{35}\}.\end{gathered} \tag{5}$$

The measure of Cohesion (Coh) is the degree to which different object types are linked. To obtain a holistic measure, three metrics are defined covering distinct perspectives: the *synchronization ratio*, the *composition index*, and *log-based co-occurrence*.

5.1.1 Synchronization Ratio

The Synchronization Ratio (SR) measures the degree of behavioral synchronization between two object types. It is defined as the Jaccard index of their transition sets. In other words, the proportion of transitions shared by two object types relative to the total number of distinct transitions that involve at least one of them. The Synchronization Ratio of two object types $ot_i, ot_j \in \mathcal{U}_{\mathrm{ot}}$ is defined as follows:

$$\mathrm{SR}(ot_i, ot_j) = \frac{|\mathrm{T}(ot_i) \cap \mathrm{T}(ot_j)|}{|\mathrm{T}(ot_i) \cup \mathrm{T}(ot_j)|} \in [0,1]. \tag{6}$$

A value close to 1 is a strong sign of behavioral cohesion. Returning to the OCPN $\mathcal{N}$, let us consider the object types CUSTOMER and ITEM. The sets of associated transitions are:

$$\begin{gathered}\mathrm{T}(\textsc{Customer}) = \{\text{place order, process payment, receive coupon, receive order}\} \\ \mathrm{T}(\textsc{Item}) = \{\text{place order, process payment, inspect item, receive order}\}\end{gathered}$$

There are three transitions in the intersection of these sets and five transitions in the union. So:

$$\mathrm{SR}(\textsc{Customer}, \textsc{Item}) = \frac{3}{5} = 0.6. \tag{7}$$

This cohesion score shows that their lifecycles are partially different. The customer may receive a coupon while the items are under inspection. In contrast, $\mathrm{SR}(\textsc{Customer}, \textsc{Merchant}) = 0$. This means that CUSTOMER and MERCHANT never synchronize on a shared activity.

The SR naturally captures the degree to which objects are behaviorally linked. However, from the perspective of process semantics, one-to-one and one-to-many relationships are fundamentally different. Treating these two situations identically leads to suboptimal clustering results. Synchronization involving a variable arc (i.e., one-to-many relationship) implies that one object instance simultaneously determines the execution of multiple instances of another type. This is a much stronger bond than the simple one-to-one interaction represented by ordinary arcs. Therefore, the Composition Index (CI) is introduced to take into account these dependencies.

5.1.2 Composition Index

The Composition Index (CI) identifies structural cohesion that comes from one-to-many relationships. It detects the presence of at least one transition t that has both a normal arc and a variable arc. Formally, the composition index of two object types $ot_i, ot_j \in \mathcal{U}_{\mathrm{ot}}$ is defined as follows in expression (8).

$$\mathtt{CI}(ot_i, ot_j) = \begin{cases} 1, & \text{if } \exists\, t \in T,\ \exists\, p_i, p_j \in P \text{ such that} \\ & \quad (p_i, t) \in F \backslash \mathcal{V}_{\text{arc}},\ (p_j, t) \in \mathcal{V}_{\text{arc}}, \\ & \quad \{\mathtt{pType}(p_i), \mathtt{pType}(p_j)\} = \{ot_i, ot_j\}. \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

In object-centric process models, many-to-many relationships are typically handled through intermediate object types [22]. Direct many-to-many synchronization via multiple variable arcs lacks clear execution semantics in the OCPN formalism [5]. For instance, a many-to-many relationship between ORDER and PACKAGE (or ROUTE) is modeled transitively through an intermediate object type ITEM as follows: $\text{ORDER} \xrightarrow{\text{one-to-many}} \text{ITEM} \xleftarrow{\text{one-to-many}} \text{PACKAGE}$. The Composition Index (CI) identifies these one-to-many structural building blocks (where $\mathtt{CI}(\text{ORDER}, \text{ITEM}) = 1$ and $\mathtt{CI}(\text{PACKAGE}, \text{ITEM}) = 1$).

The CI is deliberately defined as binary. This choice is based on the difference between the business process architecture and the execution volume. In OCPNs, a variable arc shows a fundamental structural dependency, while the number of tokens used is a runtime property. The architectural relationship between the ORDER and ITEM types stays the same, no matter how many items are in an order.

In the OCPN $\mathcal{N}$, for instance, the transition 'place order' consumes a single token from the Customer place via an ordinary arc and an arbitrary number of tokens from the Item place via a variable arc. Therefore:

$$\mathtt{CI}(\text{CUSTOMER}, \text{ITEM}) = 1. \tag{9}$$

Another example is how CUSTOMER and COUPON are related. Although they synchronize at the transition 'receive coupon', both connect via ordinary arcs. Thus, $\mathtt{CI}(\text{CUSTOMER}, \text{COUPON}) = 0$.

Note that the Synchronization Ratio and Composition Index effectively measure the theoretical potential for interaction, but they remain static model-based measures. They do not account for the frequency of real executions. Two object types may yield high SR and CI scores even if their interaction rarely occurs in reality. To bridge this gap between model structure and process execution, a data-driven metric must be added to validate the empirical significance of these relationships.

5.1.3 Log-Based Co-occurrence

The Log-Based Co-occurrence (LCO) metric provides a data-driven way to measure empirical cohesion. It uses the Jaccard index to quantify how often two object types occur in the same event. The LCO metric of two object types ot_i and ot_j is defined as:

$$\mathtt{LCO}(ot_i, ot_j) = \frac{|\mathrm{E}(ot_i) \cap \mathrm{E}(ot_j)|}{|\mathrm{E}(ot_i) \cup \mathrm{E}(ot_j)|} \in [0,1]. \tag{10}$$

An example is the object types of CUSTOMER and COUPON in the event log fragment L (see Table 1). These types co-occur only in one event, e_{35}, where customer cst_8 receives coupon cp_1. As a result, $|\mathrm{E}(\text{CUSTOMER}) \cap \mathrm{E}(\text{COUPON})| = 1$. The customers appear in a total of 25 events, whereas the coupon appears in only 1 event. So, the size of the union is $|\mathrm{E}(\text{CUSTOMER}) \cup \mathrm{E}(\text{COUPON})| = 25$. This means that:

$$\mathtt{LCO}(\text{CUSTOMER}, \text{COUPON}) = \frac{1}{25} \approx 0.04. \tag{11}$$

This low score shows that although COUPON is structurally linked to CUSTOMER in the module (via transition 'receive coupon'), their empirical bond is weak.

5.1.4 Composite Cohesion Score

This article proposes to combine the behavioral (SR), structural (CI), and empirical (LCO) metrics into a single *composite cohesion score* (Coh) to obtain a holistic measure of relational strength. This integration allows the framework to balance theoretical model dependencies with real-world execution evidence. The overall cohesion between two object types ot_i and ot_j is defined as a weighted linear combination:

$$\mathrm{Coh}(ot_i, ot_j) = \omega_1 \cdot \mathrm{SR}(ot_i, ot_j) + \omega_2 \cdot \mathrm{CI}(ot_i, ot_j) + \omega_3 \cdot \mathrm{LCO}(ot_i, ot_j), \quad (12)$$

where ω_1, ω_2, and ω_3 are user-configurable weights that determine the relative importance of each perspective. The weights satisfy:

$$\sum_{k=1}^{3} \omega_k = 1, \omega_k \in [0,1]. \quad (13)$$

In this article, an equal weighting strategy ($\omega_1 = \omega_2 = \omega_3 = 1/3$) is used as the default setting. This choice is based on the idea that behavioral synchronization, structural dependencies, and real-world evidence are equally significant indicators of semantic cohesion. It avoids biasing the decomposition toward a single perspective.

Since all constituent metrics are symmetric, $\mathrm{Coh}(ot_i, ot_j) = \mathrm{Coh}(ot_j, ot_i)$. Table 2 shows the scores calculated for the pairs of objects in the running example.

Table 2. Computed cohesion scores for object pairs in the running example

Object Pair (ot_i, ot_j)	**Behavioral** SR	**Structural** CI	**Empirical** LCO	**Composite** Coh
(Cust, Item)	0.60	1.00	0.57	**0.72**
(Cust, Coupon)	0.25	0.00	0.04	**0.10**
(Item, Merch)	0.25	0.00	0.41	**0.22**
(Cust, Merch)	0.00	0.00	0.00	**0.00**
(Item, Coupon)	0.00	0.00	0.00	**0.00**
(Coupon, Merch)	0.00	0.00	0.00	**0.00**

5.2 Clustering Object Types

Once the cohesion scores are quantified, the *Object Cohesion Graph* can be constructed to model the structural relationships between object types. This graph acts as a map of the process interactions. The nodes represent object types, and edges represent the strength of their cohesive bond. High-weight edges show object types that should likely form a single cluster. Weak edges indicate potential interfaces between clusters.

Figure 3 illustrates the Object Cohesion Graph for the running example. Note how the edge between CUSTOMER and ITEM is visually heavy ($\omega = 0.72$). It suggests a cohesive cluster. In contrast, the COUPON is only loosely attached ($\omega = 0.10$).

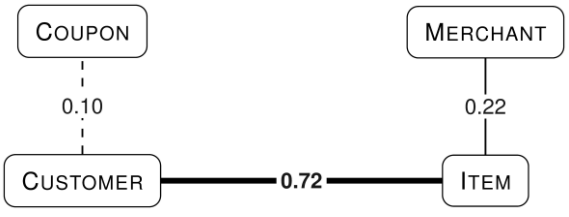

Figure 3. Object Cohesion Graph for the e-commerce running example

Formally, the Object Cohesion Graph is defined as follows in Definition 4.

Definition 4 (Object Cohesion Graph). *Let* $ON = ((P, T, F, l), \mathrm{pType}, \mathcal{V}_{\mathrm{arc}})$ *be an OCPN. The* Object Cohesion Graph *of* ON *is a weighted undirected graph* $G = (\mathcal{S}_{\mathrm{ot}}, \mathcal{E}, \omega)$*, where:*

- $\mathcal{S}_{\mathrm{ot}} = \{\mathrm{pType}(p) \mid p \in P\} \subseteq \mathcal{U}_{\mathrm{ot}}$ *is the finite set of object types within* ON*;*
- $\mathcal{E} = \{\{ot_i, ot_j\} \subseteq \mathcal{S}_{\mathrm{ot}} \mid \mathrm{Coh}(ot_i, ot_j) > 0\}$ *is a finite set of undirected edges;*
- $\omega: \mathcal{E} \to [0,1]$ *is a weight function such that, for every* $\{ot_i, ot_j\} \in \mathcal{E}$, $\omega(\{ot_i, ot_j\}) = \mathrm{Coh}(ot_i, ot_j)$.

This graph serves as input to the *Louvain community detection algorithm* [9]. It partitions the object types into disjoint clusters based on the *modularity* measure [23], [24]. Initially, each object type is assigned to its own community. The algorithm then iteratively executes two phases:

1. *Local Optimization*: For each node ot, the algorithm evaluates the local modularity gain produced by moving ot to the community of each of its neighbors. The highest positive gain is applied. This step iterates until the modularity score cannot be improved any further.
2. *Community Aggregation*: The communities detected in the first phase are collapsed into super-nodes. The edges between these new nodes are formed by summing the weights of the edges between the internal nodes of the corresponding communities.

These two phases are repeated iteratively on the reduced graph until no further positive local modularity gain can be achieved. The resolution parameter (γ) can be tuned to adjust the granularity of the resulting communities.

The output is a partition $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ of the set $\mathcal{S}_{\mathrm{ot}}$ such that $\bigcup_{i=1}^{n} \mathcal{C}_i = \mathcal{S}_{\mathrm{ot}}$ and $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset, \forall i \neq j$.

Each cluster $\mathcal{C}_i$ represents a highly cohesive group of objects that will form the basis of a sub-process module. For the running example (with a high resolution, $\gamma = 1.5$, to separate weak links), the algorithm yields three clusters:

$$\mathcal{C} = \left\{ \underbrace{\{\textsc{Customer}, \textsc{Item}\}}_{\mathcal{C}_1}, \underbrace{\{\textsc{Merchant}\}}_{\mathcal{C}_2}, \underbrace{\{\textsc{Coupon}\}}_{\mathcal{C}_3} \right\}. \tag{14}$$

By aggregating the edge weights between these clusters, the high-level structure can be visualized (Figure 4). This cluster-level view simplifies the topology. It shows that cluster $\mathcal{C}_1 = \{\textsc{Customer}, \textsc{Item}\}$ acts as the core process hub by interacting with the Merchant and Coupon clusters.

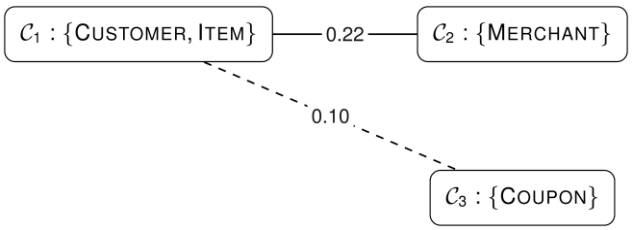

Figure 4. Visualization of the cluster interactions derived from the running example

5.3 Object-Aware Decomposition

A partition $\mathcal{C}$ that comes from the clustering phase serves as the blueprint for decomposing the model. Each cluster $\mathcal{C}_i$ is transformed into an independent sub-process module. To achieve this, the original OCPN is projected onto the identified clusters. This requires classifying the transitions based on the locations (clusters) of their associated object types.

Let $ON = ((P, T, F, l), \mathrm{pType}, \mathcal{V}_{\mathrm{arc}})$ be the original OCPN. Then it is possible to classify the set of transitions T relative to the clusters as follows:

- *Internal Transitions:* A transition t is internal to a cluster $\mathcal{C}_i$ if all its connected object types belong exclusively to that cluster. Formally:

$$T_{\mathrm{int}}^{\mathcal{C}_i} = \{\, t \in T \mid \mathrm{tType}(t) \subseteq \mathcal{C}_i \,\}. \tag{15}$$

  These transitions represent behavior that is entirely local to the module.

- *Interface Transitions:* A transition t is an interface transition if it connects object types belonging to distinct clusters. The set of interface transitions is:

$$T_{\mathrm{iface}} = T \setminus \bigcup_{\mathcal{C}_i \in \mathcal{C}} T_{\mathrm{int}}^{\mathcal{C}_i}. \tag{16}$$

  These transitions act as synchronization points between modules.

Based on this classification, the following decomposed system is defined.

Definition 5 (Decomposed OCPN System). *Let $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ be the object type partition. The* Decomposed OCPN System *is a set of modules $\mathcal{S}_{ON} = \{ON_1, \ldots, ON_n\}$. Each module ON_i is a subnet associated with cluster $\mathcal{C}_i$ and is defined as a tuple $ON_i = ((P_i, T_i, F_i, l_i), \mathrm{pType}_i, \mathcal{V}_{\mathrm{arc}}^i)$, where:*

- $P_i = \{p \in P \mid \mathrm{pType}(p) \in \mathcal{C}_i\}$ *is the set of places typed by the cluster;*
- $T_i = T_{\mathrm{int}}^{\mathcal{C}_i} \cup \{t \in T_{\mathrm{iface}} \mid \mathrm{tType}(t) \cap \mathcal{C}_i \neq \emptyset\}$ *is the set of internal transitions and relevant interface transitions;*
- $F_i = F \cap ((P_i \times T_i) \cup (T_i \times P_i))$ *is the flow relation restricted to T_i and P_i;*
- $l_i = l|_{T_i}$ *is the labeling function restricted to T_i;*
- $\mathrm{pType}_i = \mathrm{pType}|_{P_i}$ *is the place typing function restricted to P_i;*
- $\mathcal{V}_{\mathrm{arc}}^i = \mathcal{V}_{\mathrm{arc}} \cap F_i$ *is the subset of variable arcs within the module.*

Note that cohesion scores are computed specifically between object types. They are not defined between module interfaces at level 1 in the decomposed OCPN system because interfaces represent replicated synchronization transitions rather than object types.

It is important to distinguish between the structural and behavioral aspects of interface transitions. Structurally, these transitions are replicated across the connected modules. Semantically, these replicas represent a single shared synchronization event. An interface transition can only fire if it is enabled in all participating modules simultaneously.

Under this interpretation, the collection of decomposed modules maintains the behavior of the original OCPN. Internal transitions remain local to their respective modules, whereas interface transitions are execution points where modules synchronize. Consequently, the original OCPN can be reconstructed up to behavioral equivalence by synchronizing the replicated interface transitions.

Applying Definition 5 to the partition $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$, which was derived from the clustering phase in Section 5.2, yields three distinct OCPN modules ($\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$), as illustrated in Figure 5.

The *Order Fulfillment Module* ($\mathcal{N}_1$) corresponds to the core lifecycle of CUSTOMER and ITEM instances. It includes the internal transitions ‘place order’, ‘process payment’, and ‘receive order’. It also contains the interface transitions ‘inspect item’ and ‘receive coupon’ which are connected exclusively to the ITEM and CUSTOMER places, respectively. The MERCHANT and COUPON places remain invisible to this module.

The *Merchant Module* ($\mathcal{N}_2$) illustrates the behavior of the service provider. It has only the MERCHANT place and the interface transition ‘inspect item’. This represents the merchant’s view of the inspection process.

The *Coupon Module* ($\mathcal{N}_3$) separates the logic associated with the optional coupon. It consists of the COUPON place and the 'receive coupon' transition.

To enable synchronization, the interface transitions, depicted as dashed rectangles, are replicated across these modules. The flow relations (F_1, F_2, F_3) ensure that each module remains structurally self-contained while preserving the global process semantics.

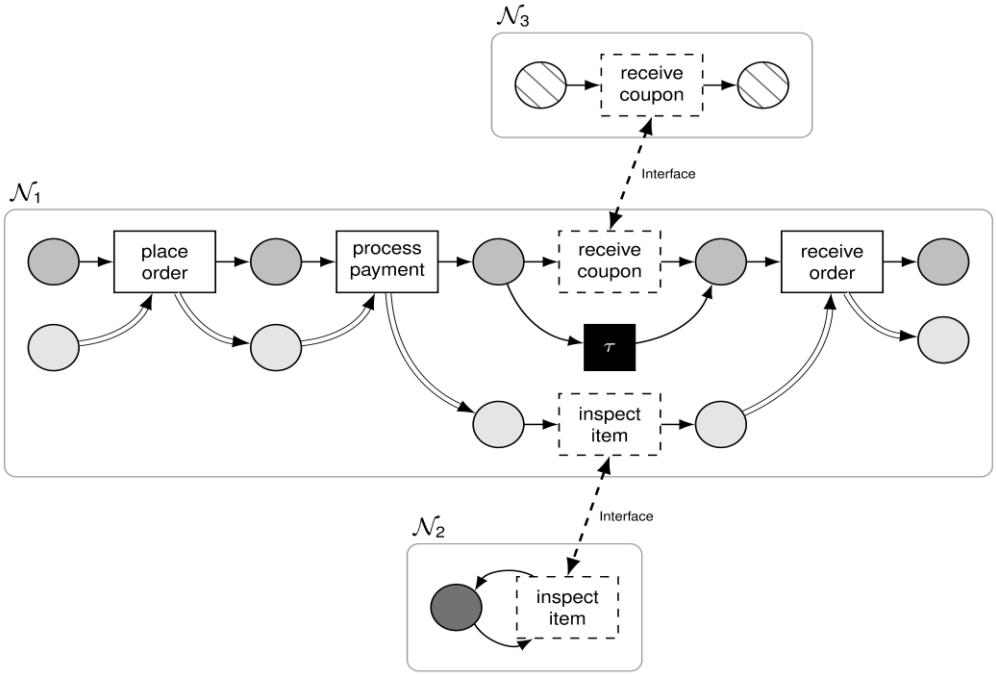

Figure 5. Decomposition of the OCPN $\mathcal{N}$ shown in Figure 2. The modules $\mathcal{N}_1$, $\mathcal{N}_2$, and $\mathcal{N}_3$ correspond to the clusters $\mathcal{C}_1 = \{\text{CUSTOMER}, \text{ITEM}\}$, $\mathcal{C}_2 = \{\text{MERCHANT}\}$, and $\mathcal{C}_3 = \{\text{COUPON}\}$, respectively.

5.4 Model Aggregation

While decomposition reduces complexity by separating cohesive sub-processes, a holistic perspective is required to comprehend the global business process architecture. To this end, the *Aggregated Net*, a high-level abstraction, is established directly from the decomposed system.

The Aggregated Net is structured as a classical Petri net. It mainly serves as a *topological map* of inter-module interactions. It abstracts away internal control-flow logic and variable arc semantics to focus exclusively on inter-module interactions. The flow relation is defined as *bidirectional*. This ensures that interface transitions act as synchronization points where modules coordinate their execution without being consumed or terminated.

Definition 6 (Aggregated Net). *Let $\mathcal{S}_{ON} = \{ON_1, \dots, ON_n\}$ be the decomposed OCPN system derived from the original net ON, and let T_{iface} be the set of interface transitions. The* Aggregated Net *is a Petri net $N_{\text{agg}} = (P_{\text{agg}}, T_{\text{agg}}, F_{\text{agg}})$, where:*

- $P_{\text{agg}} = \{ON_1, \dots, ON_n\}$ *is the set of places, where each place corresponds to a sub-process module;*
- $T_{\text{agg}} = T_{\text{iface}}$ *is the set of interface transitions;*
- F_{agg} *is the flow relation. For every module ON_i and every interface transition $t \in T_{\text{agg}}$, if t is part of ON_i, then both an input and an output arc are created between the module place and the transition. Formally:*

$$F_{\text{agg}} = \bigcup_{ON_i \in P_{\text{agg}}} \{(ON_i, t), (t, ON_i) \mid t \in T_i \cap T_{\text{agg}}\}. \tag{17}$$

In the aggregated model, a token in place ON_i means that the corresponding sub-process module is active. The bidirectional arcs create a synchronization constraint. An interface transition t can only fire if all the modules that are connected to it are active.

Figure 6 illustrates the Aggregated Net derived from the running example. It makes the structure of interaction very clear. The central module $\mathcal{N}_1$ interacts with $\mathcal{N}_2$ via 'inspect item' and with $\mathcal{N}_3$ via 'receive coupon'.

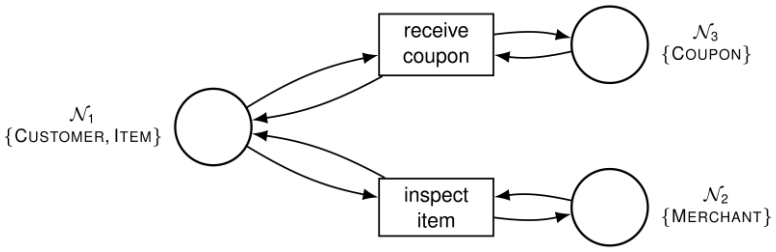

Figure 6. $\mathcal{N}_{agg}$*:* An Aggregated Net derived from $\mathcal{N}$ of Figure 2

5.5 Multi-Level Process Analysis

The original model, the decomposed system, and the aggregated abstraction form a three-level hierarchy for process analysis. This hierarchy enables moving from a map of inter-module interactions to execution-level details without losing overall context. The hierarchy can be formalized as follows.

Level 0: The Architectural View (Aggregated Net). This top level shows a clean, classical Petri net structure. It abstracts away the internal control-flow logic and the complexities of variable arcs. The analyst looks at the Aggregated Net to understand the global topology and dependencies between interacting modules. For instance, Figure 6 clearly identifies that the COUPON process is an optional side-branch of the order fulfillment.

Level 1: The Modular View (Decomposed OCPN System). The analyst "drills down" to look at the internal logic of the sub-process by choosing a macro-place at Level 0. The analyst can see the internal logic, but the external synchronization is hidden inside interface transitions. For instance, if the analyst selects the macro-place $\mathcal{N}_1$ in Figure 6 (Level 0), they will see the detailed module $\mathcal{N}_1$ in Figure 5 (Level 1). In this case, the sequence from 'place order' to 'receive order' is shown, while the synchronization with modules $\mathcal{N}_2$ and $\mathcal{N}_3$ is abstracted by the interface transitions 'inspect item' and 'receive coupon'.

Level 2: The Global Semantics (Original OCPN). While difficult to visualize for complex processes, the original OCPN remains the mathematical foundation for execution validity and global conformance checking.

6 Experimental Results and Discussion

The effectiveness of the proposed framework is evaluated using the Order Management process [25]—a well-known example from the OCEL 2.0 standard. The process includes an OCEL and its corresponding discovered OCPN. There are 21008 events, 11 activities, and 10840 objects across 6 object types in the OCEL. This dataset is suitable for evaluation because it exhibits complex interactions between objects. These include one-to-many relationships (e.g., one order having multiple items) and resource involvements (e.g., employees handling packages).

To validate the approach, a Python implementation† was developed. The tool leverages the `PM4Py` library [26] to work with object-centric event data. It also uses the `NetworkX` library and the `python-Louvain` module to construct graphs and partition the process into clusters.

† The implementation and dataset of the experiments are available at: https://github.com/khalilmecheraoui/ocpm-cohesion-decomposition.git

6.1 Cohesion Scores

The experiment begins by extracting the structural definitions from the OCPN to explicitly distinguish between ordinary arcs and variable arcs. This distinction is essential for calculating the Synchronization Ratio (SR) and the Composition Index (CI). Table 3 illustrates the configuration extracted from the Order Management process.

Table 3. Structural configuration of the Order Management OCPN. The symbol ⇑ denotes a variable arc, while ↑ denotes an ordinary arc.

Activity	Customers	Orders	Items	Products	Packages	Employees
place order	↑	↑	⇑	⇑	–	–
confirm order	↑	↑	⇑	⇑	–	↑
pay order	–	↑	⇑	⇑	–	–
payment reminder	–	↑	⇑	⇑	–	–
pick item	–	–	↑	↑	–	↑
item out of stock	–	–	↑	↑	–	↑
reorder item	–	–	↑	↑	–	↑
create package	–	–	⇑	⇑	↑	↑
send package	–	–	⇑	⇑	↑	⇑
package delivered	–	–	⇑	⇑	↑	↑
failed delivery	–	–	⇑	⇑	↑	↑

The six object types involved in the process are shown in the columns of Table 3: CUSTOMERS, ORDERS, ITEMS, PRODUCTS, PACKAGES, and EMPLOYEES. The rows of the table correspond to the process activities. The table explicitly highlights the nature of the structural connection between each activity and its associated objects.

A notable feature of this model is observed in the 'send package' activity. The latter connects to the EMPLOYEES object type through a variable arc. This indicates a complex resource interaction pattern, such as when a group of employees handles a single package.

The implemented tool calculated the pairwise cohesion scores by analyzing the Order Management event log (available through the Ocelot platform[‡]) and the structural configuration depicted in Table 3 (extracted from the corresponding object-centric Petri net) using equal weights ($\omega_1 = \omega_2 = \omega_3 = 1/3$) for the composite cohesion score. This configuration ensures a balanced contribution from behavioral, structural, and empirical perspectives. Table 4 defines the *Object Cohesion Graph* illustrated in Figure 7. In this graph, the thickness of the edges is related to the cohesion scores.

The resulting cohesion scores highlight the efficacy of the composite score in capturing diverse relationships. For instance, the pair (ITEM, PRODUCT) has a perfect cohesion score of 1.00. This indicates maximal synchronization across all perspectives. It accurately indicates that products and items are semantically inseparable in this process. Similarly, pairs involving EMPLOYEES and ITEMS or PRODUCTS received high scores (0.84). This is driven by the frequent behavioral interaction of employees with items and products.

The pair (EMPLOYEE, PACKAGE) is associated with a score of 0.57. Despite a lower log co-occurrence $\mathrm{LCO} = 0.22$, the maximal Composition Index ($\mathrm{CI} = 1.00$) successfully captured the structural assignment of employees to package processing. This demonstrates the capability of the framework to identify resource dependencies.

‡ https://ocelot.pm/

Table 4. Computed cohesion scores for all object pairs in the Order Management process. Equal weights ($\omega_k = 1/3$) were applied.

Object 1	**Object 2**	SR	CI	LCO	**Composite (Coh)**
Employees	Packages	0.50	1.00	0.22	**0.57**
Employees	Customers	0.11	0.00	0.11	**0.07**
Employees	Orders	0.09	0.00	0.10	**0.06**
Employees	Items	0.73	1.00	0.78	**0.84**
Employees	Products	0.73	1.00	0.78	**0.84**
Packages	Customers	0.00	0.00	0.00	**0.00**
Packages	Orders	0.00	0.00	0.00	**0.00**
Packages	Items	0.36	1.00	0.18	**0.51**
Packages	Products	0.36	1.00	0.18	**0.51**
Customers	Orders	0.50	0.00	0.61	**0.37**
Customers	Items	0.18	1.00	0.19	**0.46**
Customers	Products	0.18	1.00	0.19	**0.46**
Orders	Items	0.36	1.00	0.31	**0.56**
Orders	Products	0.36	1.00	0.31	**0.56**
Items	Products	1.00	1.00	1.00	**1.00**

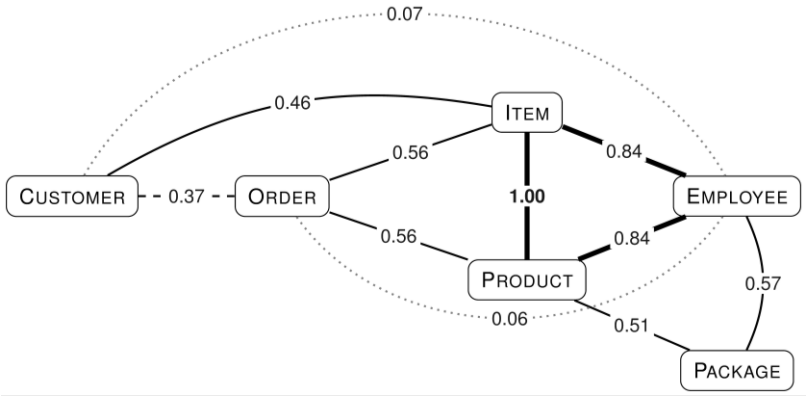

Figure 7. Object Cohesion Graph for the Order Management process. Edge thickness is strictly proportional to the cohesion score.

The pair (CUSTOMER, ORDER) received a score of 0.37. The structural composition was zero (indicating a one-to-one relationship). However, the Synchronization Ratio $\mathrm{SR} = 0.50$ and empirical evidence ($\mathrm{LCO} = 0.61$) indicated a strong commercial bond. This is sufficient to keep them clustered in the next step.

6.2 Decomposition Results

The Louvain community detection algorithm (with a standard resolution $\gamma = 1.0$) was applied to the weighted Object Cohesion Graph. This led to the object types being split into a *Logistics Cluster* $\mathcal{C}_1$ and a *Commercial Cluster* $\mathcal{C}_2$, where:

$$\mathcal{C}_1 = \{\text{PRODUCTS}, \text{ITEMS}, \text{PACKAGES}, \text{EMPLOYEES}\}$$
$$\mathcal{C}_2 = \{\text{CUSTOMERS}, \text{ORDERS}\}. \tag{18}$$

By projecting the original OCPN onto these clusters, two corresponding process modules were obtained. The *Logistics Module* (which is derived from $\mathcal{C}_1$) is highly cohesive and has most of the process logic. It consists of seven internal transitions: 'pick item', 'item out of stock', 'reorder item', 'create package', 'send package', 'package delivered', and 'failed delivery'. In addition to these internal transitions, it has four interface transitions: 'place order', 'confirm order', 'pay order', and 'payment reminder'. A significant finding is the correct grouping of EMPLOYEES with the objects they handle.

The *Commercial Module* (which is derived from $\mathcal{C}_2$) does not have any internal transitions. Instead, it synchronizes with the Logistics Module through four interface transitions: 'place order', 'confirm order', 'pay order', and 'payment reminder'.

6.3 Impact of Metric Weighting

The extreme weight configurations demonstrated the importance of the multi-perspective approach. The tool was run with high Behavioral weight ($\omega_1 = 0.9$), high Structural weight ($\omega_2 = 0.9$), and high Empirical weight ($\omega_3 = 0.9$). The weights that were not dominant stayed at 0.05. The resulting pairwise cohesion scores and clusters are shown in Table 5.

Table 5. Comparison of composite cohesion scores and clusters. C: CUSTOMERS, O: ORDERS, I: ITEMS, PR: PRODUCTS, E: EMPLOYEES, PA: PACKAGES

	Composite Cohesion (Coh)			
Pair	**Behavioral** SR ($\omega_1 = 0.9$)	**Structural** CI ($\omega_2 = 0.9$)	**Empirical** LCO ($\omega_3 = 0.9$)	**Balanced**
(Packages, Orders)	0.00	0.00	0.00	0.00
(Packages, Customers)	0.00	0.00	0.00	0.00
(Packages, Products)	0.39	0.93	0.23	0.51
(Packages, Items)	0.39	0.93	0.23	0.51
(Packages, Employees)	0.51	0.94	0.28	0.57
(Orders, Customers)	0.48	0.06	0.57	0.37
(Orders, Products)	0.39	0.93	0.35	0.56
(Orders, Items)	0.39	0.93	0.35	0.56
(Orders, Employees)	0.09	0.01	0.09	0.06
(Customers, Products)	0.22	0.92	0.23	0.46
(Customers, Items)	0.22	0.92	0.23	0.46
(Customers, Employees)	0.11	0.01	0.10	0.07
(Products, Items)	1.00	1.00	1.00	1.00
(Products, Employees)	0.74	0.98	0.79	0.84
(Items, Employees)	0.74	0.98	0.79	0.84
Clusters Found	{C, O} {E, I, PR, PA}	{C, O, I, PR} {E, PA}	{C, O} {E, I, PR, PA}	{C, O} {E, I, PR, PA}

The analysis demonstrated that relying mostly on the Synchronization Ratio ($\omega_1 = 0.9$) produced a valid partition within this particular dataset ({CUSTOMERS, ORDERS} and {ITEMS, PRODUCTS, EMPLOYEES, PACKAGES}). However, the cohesion scores are not representative of the

system because this metric lacks both the structural nuance and the empirical execution evidence required to differentiate relationships in more complex scenarios.

When the Structural Composition Index ($\mathtt{CI}$) is dominant, the implementation failed to separate the commercial objects {CUSTOMERS, ORDERS} from the logistics objects {EMPLOYEES, ITEMS, PRODUCTS, PACKAGES}. The results show that the ITEMS objects that employees physically handle are moved from the logistics cluster to the commercial cluster due to the strong variable arcs connecting them to the ORDERS objects. The resulting decomposition describes *what* the order contains, rather than *how* the process is executed.

Finally, relying only on data from the real world is not sufficient. The pair (PACKAGES, EMPLOYEES) has a low empirical cohesion of 0.28. This score is low because these objects do not always appear in the same events. The composite metric complements the low empirical score (0.22) with a behavioral score and maximal structural score ($\mathtt{CI} = 1.00$), resulting in a composite score of 0.57. This ensures that resources remain correctly bound to their modules.

This comparison shows the necessity of a multi-perspective configuration.

6.4 Aggregated Process Model

To illustrate the global topology of the Order Management process, Figure 8 shows the corresponding Aggregated Net (Level 0). This high-level abstraction represents the decomposed modules as macro-places and the interface transitions as synchronization points. It abstracts away the internal control-flow details of the underlying sub-processes.

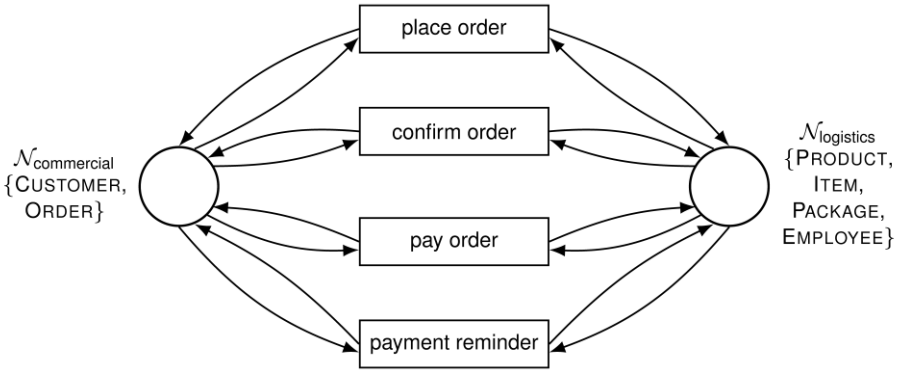

Figure 8. An Aggregated Net derived from the Order Management process

This Aggregated Net serves as a topological map. The commercial module ($\mathcal{N}_{\text{commercial}}$) synchronizes with the logistics module ($\mathcal{N}_{\text{logistics}}$) through four shared transitions. The bidirectional arcs show that both modules must be involved in the interface transitions. At this level, the order of execution (for instance, "place order" must come before "pay order") is not encoded. The internal control-flow logic in the decomposed modules (Level 1) ensures that these causal dependencies are followed. This preserves the semantics of the global process while offering a simplified architectural view.

6.5 Discussion

A primary contribution of this work is the capability to disentangle complex object interactions into semantically coherent sub-process modules while preserving the global execution context. In the Order Management benchmark evaluation, the framework successfully grouped resource objects (EMPLOYEES) with the business objects they manipulate (PACKAGES, ITEMS, and PRODUCTS). This provides a more realistic representation of the underlying system. Furthermore, the algorithm automatically identified the functional boundary between the "Commercial Front-End" (CUSTOMERS, ORDERS) and the "Logistical Back-End" (PRODUCTS, ITEMS, PACKAGES, EMPLOYEES).

This domain distinction emerged naturally from the composite cohesion metric. It validates that the combination of behavioral, structural, and empirical perspectives aligns with real-world business semantics.

The generation of the Aggregated Net (Level 0) offers a powerful abstraction mechanism. This high-level model helps analysts understand the global orchestration logic and how modules interact with each other before diving into the granular control-flow details of specific sub-processes.

The validity of the proposed framework is highlighted by the following facts:

- The generation of the Aggregated Net (Level 0) provides a significant structural simplification of the business process architecture. In the experiment, the Level 0 view simplified a model with 6 interacting object types and 11 activities into just 2 macro-module places ($\mathcal{N}_{\text{commercial}}$ and $\mathcal{N}_{\text{logistics}}$) connected via 4 interface transitions. This view effectively abstracted away internal places and ordinary control-flow arcs.
- The execution equivalence between the decomposed modules (Level 1) and the original net (Level 2) is structurally enforced through interface transition replication (Definition 5). Since interface transitions function as joint synchronization constraints that fire only when simultaneously enabled across connected modules, the execution behavior is strictly preserved.

While the evaluation on the OCEL benchmark demonstrates technical feasibility, current results rely on a standard simulation dataset. Conducting mathematical proofs represents an essential avenue for future work.

7 Conclusion

Managing the complexity of object-centric process models presents a significant challenge in process mining. This article introduces a novel framework for the automated decomposition and aggregation of Object-Centric Petri Nets (OCPNs). A Composite Cohesion Metric is introduced that balances three distinct perspectives: behavioral synchronization (SR), structural composition (CI), and log-based empirical co-occurrence (LCO). The resulting pairwise cohesion scores form an Object Cohesion Graph, which serves as input for the Louvain community detection algorithm. The latter partitions object types into logical clusters. By projecting the original OCPN onto these clusters, decomposed process modules can be obtained that provide a logical modular execution view of the system. Finally, the constructed Aggregated Net provides a high-level abstraction representing a simplified view of the global business process architecture. An analyst can use this simple net to understand how the global topology works and how different modules depend on each other.

Future work will concentrate on three main directions. First, it is necessary to extend the empirical validation of the proposed framework to large-scale, real-world enterprise event logs (e.g., from commercial ERP systems). Second, AI techniques can be used to automate the process of setting cohesion weights. This would enable the framework to adaptively optimize the scoring. Eventually, modular conformance checking support could be added to the framework. The aim is to significantly reduce computational complexity and improve the precise localization of deviations.

Data Availability Statement

The dataset analyzed during the current study is publicly available in the OCEL Standard repository: https://www.ocel-standard.org/event-logs/simulations/order-management/

References

[1] W. M. P. van der Aalst, *Process Mining: Data Science in Action*. 2nd edition, Springer, 2016. Available: https://doi.org/10.1007/978-3-662-49851-4

[2] W. M. P. van der Aalst, *Process Mining Handbook*. Lecture Notes in Business Information Processing, vol. 448. Springer, 2022. Available: https://doi.org/10.1007/978-3-031-08848-3

[3] W. M. P. van der Aalst, "Object-centric process mining: Dealing with divergence and convergence in event data," in *Software Engineering and Formal Methods. SEFM 2019. Lecture Notes in Computer Science*, vol. 11724, Springer, 2019, pp. 3–25. Available: https://doi.org/10.1007/978-3-030-30446-1_1

[4] W. M. P. van der Aalst, "Object-centric process mining: Unraveling the fabric of real processes," *Mathematics*, vol. 11, no. 12, article 2691, 2023. Available: https://doi.org/10.3390/math11122691

[5] W. M. P. van der Aalst and A. Berti, "Discovering object-centric petri nets," *Fundamenta Informaticae*, vol. 175, no. 1–4, pp. 1–40, 2020. Available: https://doi.org/10.3233/FI-2020-1951

[6] A. Berti, G. Park, M. Rafiei, and W. M. P. van der Aalst, "OCEL 2.0 specification," RWTH Aachen University, Tech. Rep., 2024. Available: https://www.ocel-standard.org/

[7] H. Dhama, "Quantitative models of cohesion and coupling in software," *Journal of Systems and Software*, vol. 29, no. 1, pp. 65–74, 1995. Available: https://doi.org/10.1016/0164-1212(94)00128-A

[8] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," *Communications of the ACM*, vol. 15, no. 12, pp. 1053–1058, 1972. Available: https://doi.org/10.1145/361598.361623

[9] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, article P10008, 2008. Available: https://doi.org/10.1088/1742-5468/2008/10/P10008

[10] W. M. P. van der Aalst, "Decomposing Petri nets for process mining: A generic approach," *Distributed and Parallel Databases*, vol. 31, pp. 471–507, 2013. Available: https://doi.org/10.1007/s10619-013-7127-5

[11] W. M. P. van der Aalst, "Decomposing process mining problems using passages," in *Application and Theory of Petri Nets. PETRI NETS 2012. Lecture Notes in Computer Science*, vol. 7347, Springer, 2012, pp. 72–91. Available: https://doi.org/10.1007/978-3-642-31131-4_5

[12] J. Muñoz-Gama, J. Carmona, and W. M. P. van der Aalst, "Conformance checking in the large: Partitioning and topology," in *Business Process Management. Lecture Notes in Computer Science*, vol. 8094, Springer, 2013, pp. 130–145. Available: https://doi.org/10.1007/978-3-642-40176-3_11

[13] H. M. W. Verbeek and W. M. P. van der Aalst, "Merging Alignments for Decomposed Replay," in *Application and Theory of Petri Nets and Concurrency. PETRI NETS 2016. Lecture Notes in Computer Science*, vol. 9698, Springer, 2016, pp. 219–239. Available: https://doi.org/10.1007/978-3-319-39086-4_14

[14] W. L. J. Lee, H. M. W. Verbeek, J. Muñoz-Gama, W. M. P. van der Aalst, and M. Sepúlveda, "Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining," *Information Sciences*, vol. 466, pp. 55–91, 2018. Available: https://doi.org/10.1016/j.ins.2018.07.026

[15] K. Mecheraoui, J. C. Carrasquel, and I. A. Lomazova, "Compositional conformance checking of nested petri nets and event logs of multi-agent systems," *arXiv preprint arXiv:2003.07291*, 2020. Available: https://doi.org/10.48550/arXiv.2003.07291

[16] M. Song, C. W. Günther, and W. M. P. van der Aalst, "Trace clustering in process mining," in *Business Process Management Workshops. BPM 2008. Lecture Notes in Business Information Processing*, vol. 17, Springer, 2009, pp. 109–120. Available: https://doi.org/10.1007/978-3-642-00328-8_11

[17] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst, "Using life cycle information in process discovery," in *Business Process Management Workshops. BPM 2015*, *Lecture Notes in Business Information Processing*, vol. 256, Springer, 2016, pp. 204–217. Available: https://doi.org/10.1007/978-3-319-42887-1_17

[18] A. Jalali, "Object Type Clustering Using Markov Directly-Follow Multigraph in Object-Centric Process Mining," *IEEE Access*, vol. 10, pp. 126569–126579, 2022. Available: https://doi.org/10.1109/ACCESS.2022.3226573

[19] S. Khayatbashi, N. Miri, and A. Jalali, "Advancing object-centric process mining with multi-dimensional data operations," *arXiv preprint arXiv:2412.00393*, 2024. Available: https://doi.org/10.48550/arXiv.2412.00393

[20] S. Khayatbashi, N. Miri, and A. Jalali, “OLAP operations for object-centric process mining,” in *Intelligent Information Systems. CAiSE 2025. Lecture Notes in Business Information Processing*, vol. 557, Springer, 2025, pp. 111–118. Available: https://doi.org/10.1007/978-3-031-94590-8_14

[21] A. Lomazova, “Nested Petri Nets – a Formalism for Specification and Verification of Multi-Agent Distributed Systems,” *Fundamenta Informaticae*, vol. 43, no. 1–4, pp. 195–214, 2000. Available: https://doi.org/10.3233/fi-2000-43123410

[22] D. Fahland, “Describing behavior of processes with many-to-many interactions,” in *Application and Theory of Petri Nets and Concurrency. PETRI NETS 2019*. Lecture Notes in Computer Science, vol. 11522, Springer, 2019, pp. 3–24. Available: https://doi.org/10.1007/978-3-030-21571-2_1

[23] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical Review E*, vol. 69, no. 2, 2004. Available: https://doi.org/10.1103/PhysRevE.69.026113

[24] M. E. J. Newman, “Modularity and community structure in networks,” *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. Available: https://doi.org/10.1073/pnas.0601602103

[25] OCEL Standard, “Order Management – OCEL 2.0 simulation log,” OCEL Standard Repository, 2023. Available: https://www.ocel-standard.org/event-logs/simulations/order-management/

[26] A. Berti, S. van Zelst, and D. Schuster, “PM4Py: A process mining library for Python,” *Software Impacts*, vol. 17, article 100556, 2023. Available: https://doi.org/10.1016/j.simpa.2023.100556