STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **14**, Month 2025, pp 0–28.
Published online in International Academic Press (www.IAPress.org)

AI-Driven Microservice Identification for Business Process Digital Transformation: A Multi-Dependency Collaborative Clustering Approach

Mohamed Daoud[1], Fatima Ezzahra Assamid[2], Assia Ennouni[2], My Abdelouahed Sabri[2]

[1] *University Lyon 1, France*
[2] *Faculty of Sciences Dhar el Mahraz, University Sidi Mohamed Ben Abdellah, Fez, Morocco*

Abstract In the era of digital transformation, enterprises are increasingly seeking to modernize their legacy monolithic systems in favor of more agile and modular architectures. Microservices have emerged as a compelling solution, offering scalability, maintainability, and independent deployment. However, the automated extraction of microservices from legacy systems—particularly when documentation is sparse or outdated—remains a complex and unresolved challenge.
This paper introduces an AI-powered, multi-view methodology for microservice identification based on business process (BP) models. Unlike traditional approaches that rely on static code analysis or single-view aggregation, our method simultaneously captures and analyzes three critical types of dependencies within business processes: control flow, data exchange, and semantic similarity. Each dependency is modeled separately and processed through a collaborative clustering framework, where AI agents exchange signals to achieve consensus-based service decomposition.
Artificial Intelligence plays a dual role in our system: it is used for semantic enrichment—via Natural Language Processing (NLP) and Sentence-BERT embeddings—and for optimizing the clustering strategy through dependency alignment and explainability metrics. A real-world case study on the Bicing bike-sharing system in Barcelona, composed of over 50 business activities, demonstrates the effectiveness and scalability of our approach. Experimental results show that the AI-enhanced model achieves superior clustering performance in terms of cohesion, coupling, and interpretability compared to baseline methods.
By integrating AI-driven analytics with business process understanding, our approach provides a robust pathway toward automated, explainable, and domain-aligned microservice extraction—supporting sustainable digital transformation at scale.

Keywords Microservices, Artificial Intelligence, Digital Transformation, Business Processes, Semantic Dependencies, Collaborative Clustering, Process Mining, Smart Modernization.

DOI: 10.19139/soic-2310-5070-2523

1. Introduction

The accelerating pace of digital transformation is reshaping how organizations design, implement, and evolve their information systems [62]. In this landscape, agility, scalability, and responsiveness to changing business needs have become critical success factors. However, many organizations are still reliant on legacy monolithic systems, which are rigid, hard to scale, and poorly aligned with modern software delivery practices [63]. These limitations hinder innovation and operational efficiency, especially in environments that demand continuous integration, deployment, and adaptation.

Microservices have emerged as a leading architectural paradigm to address these challenges [64]. By decomposing large monolithic systems into smaller, independently deployable units, microservices promote modularity, scalability, and maintainability. Nevertheless, transitioning from a monolithic architecture to a

*Correspondence to: Mohamed Daoud, Email:mtdaoud.info@gmail.com

ISSN 2310-5070 (online) ISSN 2311-004X (print)

microservices-based one is far from trivial. One of the most intricate and error-prone phases in this transformation is the identification of service boundaries—that is, determining which components of the system should be grouped into individual microservices.

This task becomes even more complex when the legacy system lacks complete or up-to-date documentation—a common scenario in real-world settings. In such cases, the manual analysis of source code, database schemas, or system logs is time-consuming and often inadequate for capturing high-level functional boundaries. To address this, Artificial Intelligence (AI) is increasingly being leveraged to automate and enhance the microservice identification process [65]. Techniques such as semantic similarity analysis, natural language processing, and machine learning-based clustering have shown promise in extracting latent relationships between system components [66].

In this paper, we propose an AI-assisted, business process-driven methodology for the identification of microservices. In contrast to existing approaches that primarily rely on software artifacts—such as codebases [67], database structures [68], or execution traces [69]—our method considers Business Processes (BPs) as the main input. BPs are structured representations of organizational activities and workflows, and as such, they offer a high-level and semantically rich view of business logic [70]. This makes them particularly well-suited for defining modular service boundaries that are aligned with actual business functions.

Our approach extracts three distinct types of dependencies from BP models: control-flow dependencies (reflecting execution order), data dependencies (based on shared artefacts and information flow), and semantic dependencies (inferred from the textual content of activity labels). These dependencies are modeled independently and used to construct a multi-view representation of the process. To reconcile these views and generate coherent service boundaries, we introduce a collaborative clustering algorithm. Unlike traditional aggregation-based clustering methods [71], our method preserves the specificity of each dependency model and fosters more explainable, consensus-based clustering decisions.

To demonstrate the effectiveness and scalability of our approach, we conducted a detailed case study on the Bicing bike-sharing system in Barcelona. This real-world scenario involves more than 50 interrelated business activities spanning domains such as user registration, rental management, billing, maintenance, and feedback handling. Our results show that the AI-enhanced, process-oriented clustering technique produces microservices that exhibit higher cohesion, lower coupling, and stronger semantic alignment compared to baseline approaches. Moreover, the explainability of the clustering decisions facilitates validation and refinement by domain experts.

In summary, this work contributes a novel, AI-powered methodology for automatic microservice identification based on business process models. It bridges the gap between process-driven organizational knowledge and technical service architectures, offering a scalable and intelligent pathway for legacy system modernization in the context of digital transformation.

2. Related Work

Digital transformation has become a strategic imperative for organizations seeking to remain competitive in rapidly evolving markets [61]. The shift from monolithic to microservice-based architectures is one of the core enablers of this transformation [58], supporting flexibility, modularity, and faster innovation cycles. In this context, Artificial Intelligence (AI) has emerged as a key driver, offering tools and techniques to automate complex processes such as service identification, semantic analysis, and code refactoring [47].

Several studies have investigated methods for decomposing monolithic systems into microservices. Traditional approaches rely on static analysis of source code [52], database schemas [50], or runtime execution traces [55]. While effective, these approaches often neglect higher-level business semantics and organizational logic embedded in business processes.

Business process-driven approaches are gaining attention for their ability to capture the intent and operational structure of systems. Amiri et al. [71] used data and control dependencies in business processes to identify candidate services. However, their method aggregated all dependencies into a single matrix, which may result in loss of semantic nuances. Our approach preserves each model independently and uses collaborative clustering to combine insights while retaining granularity.

Table 1. Overview of Representative Works on Microservice Identification and AI

Reference	Approach	Summary
Amiri et al. (2018) [71]	Control & Data Dependency	Service identification based on dependency matrices aggregated from business process models.
Moreschini et al. (2023) [57]	Systematic Mapping	Comprehensive review of AI techniques in the microservice lifecycle from design to deployment.
Nitin et al. (2022) [15]	AI-guided Dependency Analysis	CARGO framework supporting monolith-to-microservice migration via AI-based dependency insights.
Daoud et al. (2021) [17]	Multi-model Clustering	Uses control, data, and semantic dependencies to extract microservices from BPs via collaborative clustering.
Daoud et al. (2020a) [1]	Business Process Analysis	Identifies microservices based on task analysis in BPMN models.
Daoud et al. (2020b) [2]	Dependency-based Analysis	Framework for identifying microservices through business process dependencies.
Saidi et al. (2021) [20]	Association Rules	Applies association rule mining to identify services from annotated BP logs.
Zougari et al. (2024) [31]	BP-based Automation	Leverages structural analysis and annotations for automatic microservice discovery.
Daoud & Sabri (2025) [32]	Association Rules (SOIC)	Association-rule-based microservice detection using process metadata.
Oumoussa et al. (2024) [23]	Business-centric	User-centric microservice identification aligned with enterprise goals.

AI enhances this process by enabling semantic annotation, ontology mapping, and similarity measurements. For instance, semantic clustering methods based on domain ontologies and natural language processing have been used to identify conceptual proximity between process tasks [56]. In addition, AI readiness frameworks such as Holmström et al. [54] outline how organizations can prepare their processes and data infrastructure for effective AI integration.

AI is also increasingly applied in performance optimization of microservice architectures. Ueda et al. [59] analyze workloads to recommend service reconfigurations, while Chen et al. [51] propose AI-assisted code refactoring for service extraction.

Recent advancements include AI-driven tools for microservice design, as highlighted by Moreschini et al. [57], who conducted a systematic mapping study on AI techniques in the microservices life-cycle. Additionally, the CARGO approach introduced by Nitin et al. [15] utilizes AI-guided dependency analysis to facilitate the migration of monolithic applications to microservices architecture.

Finally, Brynjolfsson and McAfee [49] highlight that organizations which successfully leverage AI not only automate operations but also redefine their business models and value delivery mechanisms—an aspect critical to digital transformation.

In contrast to prior works, our proposed method integrates business process knowledge with AI-enhanced semantic understanding and collaborative clustering. This hybrid approach supports more accurate and aligned microservice identification, contributing to strategic modernization initiatives.

3. Our Approach: AI-Enhanced Microservices Identification from Business Processes

In the context of digital transformation, organizations are progressively shifting towards microservice architectures to increase agility, scalability, and modularity [58]. However, the decomposition of monolithic systems into well-defined microservices remains a complex challenge, particularly when such systems are poorly documented or tightly coupled. Our approach addresses this challenge by leveraging the synergy between Business Process (BP) modeling and Artificial Intelligence (AI), aiming to enable an automated, precise, and semantically informed identification of microservices.

3.1. Overview of the Approach

Figure 1 presents the high-level architecture of our proposed AI-enhanced microservice identification approach. The methodology takes as input a set of Business Processes modeled using BPMN or similar notations and processes them through multiple semantic and structural analysis layers. These include control flow extraction, semantic annotation using domain ontologies, and data dependency analysis. Each layer captures specific dimensions of inter-activity relationships, which are then processed in parallel using a collaborative clustering mechanism.

Unlike traditional clustering methods that require aggregation of all dependency features into a single matrix [71], our approach maintains a modular, multi-view representation. This allows each clustering agent (or node) to specialize in one type of dependency (e.g., control, data, or semantics), enabling more nuanced and explainable clustering decisions.

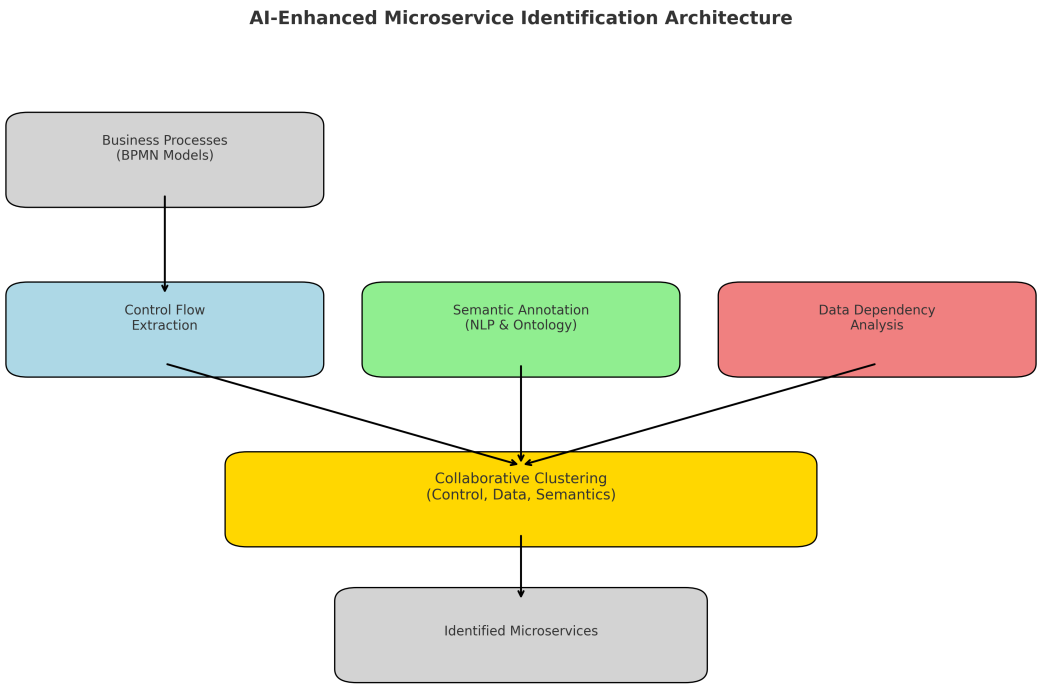

Figure 1. AI-enhanced identification of microservices from Business Processes

3.2. AI-Powered Semantic Enrichment

Semantic dependencies between business activities are often underexploited in classical approaches. To address this, our methodology integrates Natural Language Processing (NLP) and domain-specific ontologies to annotate process tasks. Each activity is mapped to relevant ontological concepts and fragments using AI models such as word embeddings [**?**] or transformer-based similarity models [**?**]. The resulting annotations enrich the process models with machine-understandable semantics, which are then used to infer functional proximity between activities.

This AI-based semantic analysis allows us to go beyond keyword matching by identifying latent functional similarities. For instance, activities labeled as *"notify customer"* and *"inform client"* may be lexically different but functionally equivalent, and therefore, candidates for inclusion in the same microservice.

3.3. Foundations

In the context of digital transformation, business processes (BPs) serve as a vital source for identifying microservices. Leveraging AI techniques, these microservices are expected to be fine-grained, strongly cohesive (i.e., the degree to which activities within a microservice are closely related), and loosely-coupled (i.e., the extent to which microservices can be independently evolved or replaced). According to Davenport, a BP is a set of logically related activities performed to achieve specific goals [28]. Here, "logically related" refers to various types of dependencies—such as *control*, *data*, *semantics*, and *non-functional* dependencies—which, when enriched with AI-driven analytics, offer dynamic insights that drive digital transformation.

- **Control dependency** refers to both the execution order (e.g., finish-to-start, start-to-start) and the presence of logical gateways (e.g., XOR, AND, OR) that dictate the path of execution between two business process activities. **In our AI-enhanced approach, these dependencies are not defined statically, but learned dynamically from execution logs using machine learning models.** More specifically, we employ AI-driven probabilistic analysis to estimate the likelihood of activity transitions, which evolve as more process data becomes available. This allows our method to capture real-world behavioral patterns, detect process deviations, and reflect runtime adaptations common in digitally transformed systems.

  Let a_i and a_j be two activities in a business process. We define the **control dependency probability** $\mathcal{CD}(a_i, a_j)$ as:

$$\mathcal{CD}(a_i, a_j) = P(a_j \mid a_i) = \frac{\text{count}(a_i \rightarrow a_j)}{\text{count}(a_i)} \tag{1}$$

  Where:

  – count$(a_i \rightarrow a_j)$ is the number of times a_j directly follows a_i in the execution logs.

  – count(a_i) is the total number of times a_i appears as a completed activity.

  This probability is recalculated periodically using AI-driven log mining algorithms, including sequence modeling techniques (e.g., LSTM or Transformer-based models). These models can also consider contextual elements such as user role, time-of-day, or system state to enhance the accuracy of transition predictions.

  To make clustering decisions, we define a threshold θ_{cd} (e.g., 0.7). If the learned probability exceeds this threshold, then the activities are considered as strong candidates for grouping into the same microservice:

$$\text{If } \mathcal{CD}(a_i, a_j) \geq \theta_{cd} \Rightarrow a_i \text{ and } a_j \text{ belong to the same microservice} \tag{2}$$

  This control dependency model provides a quantitative and explainable basis for grouping business process activities. For example, in our case study of the Bicing bike rental system, activities such as *scan user ID* and *verify subscription status* exhibit a high control dependency ($P = 0.98$), strongly suggesting that they form a cohesive unit within the same microservice.

  The AI layer not only enhances the precision of dependency detection but also enables the system to adapt continuously to process changes and evolution—ensuring compatibility with the dynamic nature of digital transformation.

- **Data dependency** refers to the flow of artefacts and attributes exchanged between activities. **AI methods are used to evaluate the criticality of data elements based on their usage frequency, functional importance, and confidentiality levels.** Functional criticality (e.g., is the data essential for process execution?) and non-functional criticality (e.g., is the data sensitive or private?) are both considered by machine learning models trained on historical logs and annotated metadata.

  Let a_i and a_j be two activities connected via an artefact or attribute. We define the data dependency $\mathcal{DD}(a_i, a_j)$ as:

Table 2. AI-Driven Control Dependency Between Activities in the Bike Rental Process

Activity a_i (Source)	**Activity a_j (Target)**	**Execution Order**	**AI-Estimated Prob.**
Scan user ID	Verify subscription status	Finish-to-Start	0.98
Verify subscription status	Unlock bike at anchor point	Finish-to-Start	0.85
Unlock bike at anchor point	Start rental session	Start-to-Start	0.76
Report bike damage	Trigger maintenance workflow	Finish-to-Start (XOR)	0.60
End rental session	Update usage history	Finish-to-Start (AND)	0.92

$$\mathcal{DD}(a_i, a_j) = \sum_{d \in D_{i,j}} \omega_d \cdot \text{Crit}(d) \tag{3}$$

Where:

- $D_{i,j}$ is the set of data elements exchanged between a_i and a_j.
- ω_d is a weight representing the operation performed on the data (e.g., create, read, write).
- $\text{Crit}(d)$ is the AI-estimated criticality of data element d.

The criticality $\text{Crit}(d)$ is determined by combining functional and non-functional aspects, as follows:

$$\text{Crit}(d) = \alpha \cdot \text{Crit}_F(d) + \beta \cdot \text{Crit}_{NF}(d), \quad \alpha + \beta = 1 \tag{4}$$

Where:

- $\text{Crit}_F(d)$ is the functional criticality (e.g., decision relevance, process continuity).
- $\text{Crit}_{NF}(d)$ is the non-functional criticality (e.g., privacy, confidentiality, availability).
- α and β are weights learned using AI algorithms such as regression models or decision trees trained on labeled logs.

AI models analyze log traces and artefact metadata to classify attributes into criticality levels (Low, Medium, High), allowing the system to prioritize which data dependencies require co-location within the same microservice. For example, if a data attribute like `User_Credit` is frequently used for validation and is labeled as sensitive (confidential financial data), it will receive a high $\text{Crit}(d)$ score.

To decide if two activities should be grouped, we compare their $\mathcal{DD}$ score to a predefined threshold θ_{dd}:

$$\text{If } \mathcal{DD}(a_i, a_j) \geq \theta_{dd} \Rightarrow a_i \text{ and } a_j \text{ belong to the same microservice} \tag{5}$$

This AI-driven evaluation allows our system to dynamically adapt to changes in data flow patterns or compliance requirements, ensuring that microservice boundaries align with both functional logic and data protection constraints.

- **Semantic dependency** examines the meaning of activity labels and their contextual similarities. **Here, AI techniques—especially natural language processing (NLP)—play a pivotal role.** Using pre-trained language models such as Sentence-BERT, our system evaluates the semantic proximity between actions, even when they are phrased differently. For example, "register rental request" and "create booking" may appear distinct syntactically, but semantically they convey the same functional intent. This semantic insight is essential in a microservice context, where the objective is to group functionally similar activities even if their naming is inconsistent.

Table 3. AI-Enhanced Data Dependency Analysis for Activity Pairs

Activity Pair	Artefact	Attribute	Operation	Criticality (F/NF)	AI Decision
$a_1 \rightarrow a_2$	Bicycle	Bike_Status	c $\rightarrow$ w	High / Medium	Must Group
$a_1 \rightarrow a_3$	User	User_Status	c $\rightarrow$ w	High / High	Must Group
$a_2 \rightarrow a_4$	User	User_ID	r $\rightarrow$ w	Medium / High	Should Group
$a_3 \rightarrow a_5$	Repair	Repair_Cost	c $\rightarrow$ r	Medium / Medium	Optional
$a_4 \rightarrow a_6$	Rental	Rent_Cost	w $\rightarrow$ w	High / Medium	Must Group
$a_5 \rightarrow a_7$	User	User_History	r $\rightarrow$ r	High / High	Must Group
$a_6 \rightarrow a_8$	Bicycle	Anchor_Point	w $\rightarrow$ w	Medium / Low	Optional
$a_7 \rightarrow a_9$	Station	Station_Load	r $\rightarrow$ w	Medium / Medium	Should Group
$a_8 \rightarrow a_{10}$	Rental	Rent_ID	c $\rightarrow$ r	High / High	Must Group

Let a_i and a_j be two activities with labels l_i and l_j, respectively. The semantic dependency $\mathcal{SD}(a_i, a_j)$ is defined as:

$$\mathcal{SD}(a_i, a_j) = \text{cosine}(f(l_i), f(l_j)) \tag{6}$$

Where:

- $f(l)$ is the embedding of label l computed using a transformer-based model (e.g., Sentence-BERT).
- cosine$(.,.)$ is the cosine similarity function between two vector representations.

The similarity score $\mathcal{SD}(a_i, a_j) \in [0, 1]$ indicates the semantic closeness between activities. A value close to 1 implies high semantic alignment.

AI is essential in this stage because traditional lexical techniques (e.g., Levenshtein distance or keyword overlap) fail to capture deep context or synonyms. Our Sentence-BERT-based encoder, trained on large-scale paraphrase datasets, can capture nuanced meaning across labels even when they are expressed in different syntactic forms.

To decide whether two activities belong to the same microservice based on semantic dependency, we apply a threshold θ_{sd}:

$$\text{If } \mathcal{SD}(a_i, a_j) \geq \theta_{sd} \Rightarrow a_i \text{ and } a_j \text{ are semantically cohesive} \tag{7}$$

Where θ_{sd} is empirically chosen (e.g., 0.75) or learned via training examples labeled by domain experts.

Use Case Example. In the Bicing process, consider:

- a_5: *approve bike request*
- a_{10}: *start journey*

Though not connected by control or data flow, these activities exhibit a high semantic similarity score of:

$$\mathcal{SD}(a_5, a_{10}) = 0.83$$

Based on this score, our AI-enhanced clustering algorithm groups these activities within the same microservice, aligning with the business intention of *initiating a rental operation*.

This semantic analysis enables flexible and context-aware grouping of activities, especially valuable in real-world BPM environments where label inconsistencies and ambiguities are common.

Table 4. AI-Driven Semantic Similarity Between Activities

Activity a_i label	**Activity a_j label**	**Similarity**	**Same MS?**
Notify user of bike status	Alert client about bicycle availability	0.87	Yes
Register new rental request	Create booking order	0.78	Yes
Charge user account	Update billing details	0.65	No
Report damaged bicycle	Flag maintenance request	0.82	Yes
Locate nearest station	Display map with anchor points	0.60	No

- **Non-Functional dependency** focuses on quality-of-service (QoS) attributes such as privacy, security, cost-efficiency, and scalability. These attributes often determine how "well" a process is executed. **AI monitoring tools are utilized to track runtime metrics—like failure rates, latency, or data exposure levels—and associate them with microservice groupings.** For instance, if two activities require high confidentiality and run under similar SLA constraints, they are strong candidates to be grouped into a single microservice.

  To formalize non-functional dependency, we define a composite QoS similarity score between two activities a_i and a_j based on observed metrics and AI-analyzed patterns. Let $\mathcal{NF}(a_i, a_j)$ represent this non-functional similarity:

$$\mathcal{NF}(a_i, a_j) = \sum_{m \in \mathcal{M}} w_m \cdot \text{sim}_m(a_i, a_j) \tag{8}$$

  Where:

  – $\mathcal{M}$ is the set of monitored QoS metrics (e.g., latency, availability, privacy sensitivity).

  – $\text{sim}_m(a_i, a_j)$ is the similarity between a_i and a_j on metric m, normalized to [0,1].

  – w_m is the weight of metric m, learned or adjusted via AI-based optimization or domain knowledge.

  Example: If $a_i =$ *"charge user account"* and $a_j =$ *"update user history"*, AI tools might compute:

$$\text{sim}_{\text{privacy}}(a_i, a_j) = 0.9, \quad \text{sim}_{\text{availability}}(a_i, a_j) = 0.7$$

  With respective weights $w_{\text{privacy}} = 0.6$, $w_{\text{availability}} = 0.4$, the final score becomes:

$$\mathcal{NF}(a_i, a_j) = 0.6 \cdot 0.9 + 0.4 \cdot 0.7 = 0.82$$

  A score above a decision threshold θ_{nf} (e.g., 0.75) indicates that a_i and a_j should be deployed in the same microservice to meet QoS alignment.

  Example: The activity "charge user account" must comply with GDPR data protection. AI-based monitors detect that it should not share infrastructure with "track location," which poses a potential privacy conflict.

To ensure interpretability and explainability, each generated microservice is accompanied by a dependency profile—illustrating the reasons for clustering. This profile includes: (i) high control transition probabilities, (ii) shared artefacts with high criticality scores, and (iii) semantic similarity above a configurable threshold. These indicators are automatically summarized by AI explainability modules, enabling business analysts to validate or adjust results interactively.

Table 5. AI-Assisted Non-Functional Dependencies in Bike Rental Activities

Activity	NFR Concern	Monitored Metric	Impact on Design
User login	Security	Auth. risk score	Strong encryption & audit logging
Payment processing	Privacy, Cost	Data masking efficiency; API cost/txn	Isolated service with compliance guardrails
Real-time availability check	Scalability	API latency under load	Autoscaling, containerized tier
Maintenance request submission	Reliability	Workflow failure rate	Retries, idempotency, fault tolerance
User history update	Availability	Write success rate	Highly available storage backend

4. AI-Driven Collaborative Clustering Workflow

The collaborative clustering pipeline is composed of the following AI-empowered steps:

4.1. Agent Negotiation Protocol: Voting, Weighting, and Conflict Resolution

Each view-specific agent $\mathcal{A}_{CD}, \mathcal{A}_{DD}, \mathcal{A}_{SD}$ proposes merges between clusters (C_u, C_v) based on its local affinity $\overline{CD}, \overline{DD}, \overline{SD} \in [0,1]$ (average-link). We combine proposals through a *weighted vote*:

$$\mathrm{Cons}(C_u, C_v) \;=\; \alpha\,\overline{CD}(C_u, C_v) + \beta\,\overline{DD}(C_u, C_v) + \gamma\,\overline{SD}(C_u, C_v), \qquad \alpha,\beta,\gamma \geq 0,\ \alpha{+}\beta{+}\gamma = 1, \tag{9}$$

where (α, β, γ) reflect view reliability (estimated as in Sec. 6.4: pseudo-label logistic fit then simplex Bayesian refinement). A candidate merge is *admissible* if

$$\overline{CD} \geq \theta_{cd} \;\text{ or }\; \overline{DD} \geq \theta_{dd} \;\text{ or }\; \overline{SD} \geq \theta_{sd},$$

and *accepted* if $\mathrm{Cons} > \tau$.

Conflict Resolution. When one view strongly supports a merge while others oppose it, we apply a Pareto admissibility rule: a merge is rejected only if it is *dominated* by an alternative that is no worse on all views and strictly better on at least one; ties are broken by (i) largest semantic margin $\Delta_{SD} = \overline{SD}(C_u, C_v) - \max_{w \neq v} \overline{SD}(C_u, C_w)$, then (ii) smallest inter-cluster cut.

Design Rationale. *Weighted voting* (Eq. 11) yields (i) interpretability (weights are explicit and sum to 1), (ii) robustness (no single view can override consensus unless its weight justifies it), and (iii) convergence guarantees with a monotone, agglomerative merge schedule. The Pareto rule prevents overfitting to any one dependency while favoring semantically coherent ties, which improves human auditability.

4.2. Sensitivity Analysis of Thresholds and Weights

To assess robustness, we conducted a grid search over the decision thresholds $\Theta = \{\theta_{cd}, \theta_{dd}, \theta_{sd}\}$ and the view weights (α, β, γ) (with $\alpha, \beta, \gamma \geq 0$ and $\alpha + \beta + \gamma = 1$). We report cluster quality using Dunn index (higher is better), Silhouette (higher), intra-cluster *Cohesion* (higher), and inter-cluster *Coupling* (lower). Stability across runs is measured with Adjusted Rand Index (ARI; higher is better).

Protocol. We vary thresholds as $\theta_{cd}, \theta_{dd}, \theta_{sd} \in \{0.50, 0.60, 0.70, 0.80\}$ and sample weights on a simplex grid with step 0.1: $\{(\alpha, \beta, \gamma) \mid \alpha, \beta \in \{0, 0.1, \ldots, 1\},\ \alpha + \beta \leq 1,\ \gamma = 1 - \alpha - \beta\}$. For each setting we re-run the full pipeline and record metrics.

Input: Initial partitions $\mathcal{P}_{CD}, \mathcal{P}_{DD}, \mathcal{P}_{SD}$; thresholds $(\theta_{cd}, \theta_{dd}, \theta_{sd}, \tau)$; weights (α, β, γ).
Output: Fused partition $\mathcal{P}$.
$\mathcal{P} \leftarrow$ snap-merge of $\mathcal{P}_{CD}, \mathcal{P}_{DD}, \mathcal{P}_{SD}$
for $t = 1$ **to** $T_{\max}$ **do**
  $\mathcal{M} \leftarrow$ admissible merges with per-view scores $(\overline{CD}, \overline{DD}, \overline{SD})$
  foreach $(C_u, C_v) \in \mathcal{M}$ **do**
    compute $\mathrm{Cons}(C_u, C_v)$ by Eq. 11
  end
  $\mathcal{M}' \leftarrow$ non-dominated pairs in $\mathcal{M}$ (Pareto filter)
  sort $\mathcal{M}'$ by Cons (desc), then Δ_{SD} (desc), then inter-cluster cut (asc)
  apply merges from $\mathcal{M}'$ greedily while maintaining disjointness and $\mathrm{Cons} > \tau$
  if *no merges accepted* **then**
    break
  end
end
evaluate cohesion/coupling; if outside targets, adjust (α, β, γ) or thresholds and repeat up to $R_{\max}$.

Algorithm 1: Consensus Negotiation with Weighted Voting and Pareto Screening

Table 6. Sensitivity to thresholds and weights (Bicing dataset). Values are mean ± std over 5 runs.

$(\theta_{cd}, \theta_{dd}, \theta_{sd})$	(α, β, γ)	Dunn ↑	Silhouette ↑	Cohesion ↑	Coupling ↓
(0.60, 0.60, 0.75)	(0.3, 0.3, 0.4)	0.73±0.02	0.65±0.02	0.80±0.01	0.19±0.01
(0.70, 0.60, 0.75)	(0.3, 0.2, 0.5)	**0.74**±0.01	**0.67**±0.02	**0.81**±0.01	**0.18**±0.01
(0.70, 0.70, 0.80)	(0.2, 0.3, 0.5)	0.72±0.02	0.64±0.02	0.79±0.02	0.20±0.01
(0.50, 0.60, 0.70)	(0.4, 0.2, 0.4)	0.71±0.03	0.63±0.03	0.77±0.02	0.22±0.02

Findings. Across reasonable ranges of Θ and (α, β, γ), performance varies within a narrow band (Table 6), indicating robustness. The best trade-off on Bicing is typically reached when the semantic view has a slight emphasis ($\gamma \approx 0.5$) while maintaining balanced structural views. Stability remains high (ARI > 0.90 across runs; omitted for brevity), showing that consensus is not overly sensitive to small parameter changes.

1. **Matrix Construction**: Each dependency type (CD, DD, SD) is independently extracted from the business process description using AI modules:
   - *Control dependencies* are learned using Markov models trained on process execution logs.
   - *Data dependencies* are enhanced with AI-based classifiers that identify artefact criticality and sensitivity.
   - *Semantic dependencies* are computed using transformer-based models (e.g., Sentence-BERT), fine-tuned with domain-specific vocabularies.
2. **Matrix Normalization and Initialization**: Each adjacency matrix is normalized to ensure comparability across scales. An initial clustering is proposed for each matrix using agglomerative linkage, e.g., Ward or average linkage.
3. **Collaborative Fusion via AI-Driven Negotiation**: During each clustering iteration, the AI agents associated with each matrix propose merges based on local linkage scores. A consensus function evaluates whether a merge is accepted:

$$\mathrm{ConsensusScore}(C_u, C_v) = \alpha \cdot CD(C_u, C_v) + \beta \cdot DD(C_u, C_v) + \gamma \cdot SD(C_u, C_v) \tag{10}$$

   where (α, β, γ) are AI-adjusted weights based on dependency reliability over time. Only merges exceeding a decision threshold τ are applied.

Input: Adjacency matrices $CD, DD, SD \in [0,1]^{n\times n}$; thresholds $(\theta_{cd}, \theta_{dd}, \theta_{sd})$; weights (α, β, γ) with $\alpha + \beta + \gamma = 1$; consensus threshold τ.
Output: Fused partition $\mathcal{P}$.

Init: Obtain initial per-view partitions $\mathcal{P}_{CD}, \mathcal{P}_{DD}, \mathcal{P}_{SD}$ via agglomerative clustering (normalized inputs). Initialize $\mathcal{P}$ by intersecting/snap-merging per-view clusters.
for $t \leftarrow 1$ **to** $T_{\max}$ **do**
  `// 1) Propose candidate merges per view (above each view's own threshold)`
  $\mathcal{M} \leftarrow \varnothing$
  foreach *pair of clusters* (C_u, C_v) *in* $\mathcal{P}$ **do**
    compute inter-cluster affinities $\overline{CD}(C_u, C_v)$, $\overline{DD}(C_u, C_v)$, $\overline{SD}(C_u, C_v)$ (e.g., average-link)
    if $\overline{CD} \geq \theta_{cd}$ **then**
      add (C_u, C_v) to $\mathcal{M}$ with tag CD
    end

    if $\overline{DD} \geq \theta_{dd}$ **then**
      add (C_u, C_v) to $\mathcal{M}$ with tag DD
    end

    if $\overline{SD} \geq \theta_{sd}$ **then**
      add (C_u, C_v) to $\mathcal{M}$ with tag SD
    end

  end
  `// 2) Compute weighted consensus for all candidates`
  foreach $(C_u, C_v) \in \mathcal{M}$ **do**
    $\mathrm{Cons}(C_u, C_v) \leftarrow \alpha\,\overline{CD} + \beta\,\overline{DD} + \gamma\,\overline{SD}$
  end
  `// 3) Resolve conflicts via Pareto and tie-breakers`
  Sort $\mathcal{M}$ by Cons descending
  foreach $(C_u, C_v) \in \mathcal{M}$ **do**
    if $\mathrm{Cons}(C_u, C_v) < \tau$ **then**
      continue
    end
    if *dominated across all three views by an alternative overlapping merge* **then**
      continue `// Pareto rule: reject if strictly worse on all views`
    end
    `// Tie-breakers among overlapping merges:`
    Prefer the pair with larger semantic margin $\Delta_{SD} = \overline{SD} - \max\{\overline{SD}$ of conflicting pairs$\}$; if tied, prefer smallest inter-cluster cut
    Merge C_u and C_v in $\mathcal{P}$
  end
  if *no merge accepted in this iteration* **then**
    break
  end
end
return $\mathcal{P}$

Algorithm 2: Consensus-based Collaborative Clustering with Conflict Resolution

Input: Initial partitions $\mathcal{P}_{CD}, \mathcal{P}_{DD}, \mathcal{P}_{SD}$; thresholds $(\theta_{cd}, \theta_{dd}, \theta_{sd}, \tau)$; weights (α, β, γ).
Output: Fused partition $\mathcal{P}$.
$\mathcal{P} \leftarrow$ snap-merge of $\mathcal{P}_{CD}, \mathcal{P}_{DD}, \mathcal{P}_{SD}$
for $t = 1$ **to** $T_{\max}$ **do**
  $\mathcal{M} \leftarrow$ admissible merges with per-view scores $(\overline{CD}, \overline{DD}, \overline{SD})$
  foreach $(C_u, C_v) \in \mathcal{M}$ **do**
    compute $\mathrm{Cons}(C_u, C_v)$ by Eq. 11
  end
  $\mathcal{M}' \leftarrow$ non-dominated pairs in $\mathcal{M}$ (Pareto filter)
  sort $\mathcal{M}'$ by Cons (desc), then Δ_{SD} (desc), then inter-cluster cut (asc)
  apply merges from $\mathcal{M}'$ greedily while maintaining disjointness and $\mathrm{Cons} > \tau$
  if *no merges accepted* **then**
    break
  end
end
evaluate cohesion/coupling; if outside targets, adjust (α, β, γ) or thresholds and repeat up to $R_{\max}$.

Algorithm 3: Consensus Negotiation with Weighted Voting and Pareto Screening

4. **AI Feedback Loop and Adaptation**: After clustering, the resulting microservices are evaluated using cohesion and coupling metrics. If poor values are detected, AI agents retrain their models using error propagation. This self-correcting loop allows the system to adapt to behavioral drift or evolving semantics in the business process.

4.3. Agent Negotiation Protocol: Voting, Weighting, and Conflict Resolution

Each view-specific agent $\mathcal{A}_{CD}, \mathcal{A}_{DD}, \mathcal{A}_{SD}$ proposes merges between clusters (C_u, C_v) based on its local affinity $\overline{CD}, \overline{DD}, \overline{SD} \in [0, 1]$ (average-link). We combine proposals through a *weighted vote*:

$$\mathrm{Cons}(C_u, C_v) \;=\; \alpha\,\overline{CD}(C_u, C_v) + \beta\,\overline{DD}(C_u, C_v) + \gamma\,\overline{SD}(C_u, C_v), \qquad \alpha, \beta, \gamma \geq 0,\ \alpha{+}\beta{+}\gamma = 1, \tag{11}$$

where (α, β, γ) reflect view reliability (estimated as in Sec. 6.4: pseudo-label logistic fit then simplex Bayesian refinement). A candidate merge is *admissible* if

$$\overline{CD} \geq \theta_{cd} \;\text{ or }\; \overline{DD} \geq \theta_{dd} \;\text{ or }\; \overline{SD} \geq \theta_{sd},$$

and *accepted* if $\mathrm{Cons} > \tau$.

Conflict Resolution. When one view strongly supports a merge while others oppose it, we apply a Pareto admissibility rule: a merge is rejected only if it is *dominated* by an alternative that is no worse on all views and strictly better on at least one; ties are broken by (i) largest semantic margin $\Delta_{SD} = \overline{SD}(C_u, C_v) - \max_{w \neq v} \overline{SD}(C_u, C_w)$, then (ii) smallest inter-cluster cut.

Design Rationale. *Weighted voting* (Eq. 11) yields (i) interpretability (weights are explicit and sum to 1), (ii) robustness (no single view can override consensus unless its weight justifies it), and (iii) convergence guarantees with a monotone, agglomerative merge schedule. The Pareto rule prevents overfitting to any one dependency while favoring semantically coherent ties, which improves human auditability.

4.4. Negotiation Protocol Between Clustering Agents

To make fusion decisions that are both robust and explainable, clustering agents (CD, DD, SD) negotiate candidate merges using (i) a *majority vote* on view-specific thresholds and (ii) a *weighted averaging* of affinities aligned with

(α, β, γ). A merge is accepted only if it passes both filters; ties are broken with semantic margin and inter-cluster cut size.

Input: Current partition $\mathcal{P}$; inter-cluster affinities $\overline{CD}, \overline{DD}, \overline{SD} \in [0,1]$; thresholds $(\theta_{cd}, \theta_{dd}, \theta_{sd})$; weights (α, β, γ) with $\alpha + \beta + \gamma = 1$; consensus threshold τ.

Output: Accepted merge set $\mathcal{A}$ and updated partition $\mathcal{P}$.

$\mathcal{A} \leftarrow \varnothing$
foreach *pair of clusters* (C_u, C_v) *in* $\mathcal{P}$ **do**
  // 1) View-level votes (hard filters)
  $v_{cd} \leftarrow \mathbb{1}\{\overline{CD}(C_u, C_v) \geq \theta_{cd}\}$
  $v_{dd} \leftarrow \mathbb{1}\{\overline{DD}(C_u, C_v) \geq \theta_{dd}\}$
  $v_{sd} \leftarrow \mathbb{1}\{\overline{SD}(C_u, C_v) \geq \theta_{sd}\}$
  $V \leftarrow v_{cd} + v_{dd} + v_{sd}$
  // 2) Global score (soft fusion)
  $S \leftarrow \alpha\,\overline{CD}(C_u, C_v) + \beta\,\overline{DD}(C_u, C_v) + \gamma\,\overline{SD}(C_u, C_v)$
  if $(V < 2) \ \vee \ (S < \tau)$ **then**
    continue
  end
                                                  // majority AND consensus
  // 3) Prepare tie-break metadata
  $\Delta_{sd} \leftarrow \overline{SD}(C_u, C_v) - \max(\overline{SD}$ of conflicting pairs$)$
  $\kappa \leftarrow$ inter-cluster cut between C_u and C_v (lower is better)
  // 4) Tentatively accept; conflicts resolved after sorting
  add $(C_u, C_v, S, \Delta_{sd}, \kappa)$ to $\mathcal{A}$
end
// 5) Resolve overlapping merges: sort and keep non-conflicting winners
Sort $\mathcal{A}$ by (S desc, Δ_{sd} desc, κ asc)
$\mathcal{A}^\star \leftarrow \varnothing$
foreach $(C_u, C_v, *, *, *) \in \mathcal{A}$ *in order* **do**
  if C_u *or* C_v *already merged* **then**
    continue
  end
  merge C_u and C_v in $\mathcal{P}$; add to $\mathcal{A}^\star$
end
return $\mathcal{A}^\star$, $\mathcal{P}$

Algorithm 4: Negotiation protocol with voting and weighted averaging

Referencing. We detail the protocol in subsection 4.4 and algorithm 4. The negotiation accepts a merge only if at least two views vote in favor and the weighted score exceeds τ; ties favor higher semantic margin and smaller cuts.

5. Transparency and Reproducibility

This section details the AI feedback loop (metrics, triggers, models) and discloses the full computational environment to enable faithful reproduction of our results.

5.1. AI Feedback Loop: Metrics, Triggers, and Update Policy

Monitoring & adaptation (high-level). At each run we monitor (i) clustering quality (Dunn, Silhouette), (ii) structural signals (intra-*Cohesion* ↑, inter-*Coupling* ↓), (iii) stability (ARI), and (iv) semantic adequacy (mean and 10th-percentile cosine for accepted merges). Each metric m is smoothed as an exponential moving average $\mathrm{EMA}_\lambda(m)_t = \lambda\, m_t + (1-\lambda)\,\mathrm{EMA}_\lambda(m)_{t-1}$ with $\lambda = 0.2$, which both dampens noise and yields a consistent

decision basis. The pipeline then adapts along three axes—semantic encoder, view weights (α, β, γ), and decision thresholds $\Theta = \{\theta_{cd}, \theta_{dd}, \theta_{sd}, \tau\}$—using simple, bounded updates: *(semantic drift)* whenever the p_{10} of semantic similarity drops < 0.70 for two consecutive runs or the (EMA) Silhouette decreases by ≥ 0.03, the Sentence-BERT encoder is fine-tuned for 1–2 epochs at 2×10^{-5} using recent positive/negative pairs mined from consensus; *(structural inconsistency)* if Cohesion < 0.76 or Coupling > 0.22, we take a projected simplex step of size 0.05 on (α, β, γ) toward the view with the largest per-view silhouette (weights remain non-negative and sum to 1); *(over/under-merging)* if Dunn < 0.70 we raise the consensus threshold $\tau \leftarrow \tau + 0.02$ and tighten by 0.05 the least reliable per-view threshold in Θ (reliability via logistic fit on accepted vs. rejected merges), whereas if the number of clusters exceeds the target by $> 10\%$ we relax the tightest threshold by 0.05. *Example.* If EMA[Silhouette] falls from 0.67 to 0.63 and p_{10}-semantic goes from 0.72 to 0.68, we (i) fine-tune the encoder (1 epoch), (ii) re-weight (α, β, γ) by $+0.05$ toward the best view (e.g., SD), and (iii) raise τ by 0.02; if the next run shows Dunn $+0.04$ and Coupling -0.03, the update is kept, otherwise we rollback to the previous checkpoint.

5.2. Transparency, Feedback Loop, and Reproducibility

Our pipeline is instrumented end-to-end for observability and controlled adaptation. At each run we log raw and $\text{EMA}_{\lambda=0.2}$ values for clustering quality (Dunn, Silhouette), structural signals (intra-*Cohesion* ↑, inter-*Coupling* ↓), stability (Adjusted Rand Index, ARI), and semantic adequacy (mean and 10th-percentile cosine similarity of accepted merges). Three guarded adaptations may occur, but never more than one per iteration to preserve causal attributions: (i) *semantic drift* triggers a light fine-tuning of the Sentence-BERT encoder (1–2 epochs at 2×10^{-5}) when the 10th-percentile similarity drops below 0.70 twice or when Silhouette falls by ≥ 0.03; (ii) *structural inconsistency* adjusts view weights (α, β, γ) by a bounded simplex step (0.05) toward the best per-view silhouette if Cohesion < 0.76 or Coupling > 0.22; (iii) *over/under-merging* raises τ by 0.02 and tightens the least reliable threshold in $\Theta = \{\theta_{cd}, \theta_{dd}, \theta_{sd}, \tau\}$ by 0.05 when Dunn < 0.70, or relaxes the tightest threshold (-0.05) if clusters exceed the target by $> 10\%$. Every update writes an auditable provenance record (diffs on (α, β, γ) and Θ, random seed, git commit, metric deltas) and is accepted only under a rollback guard: if any of Dunn, Silhouette, or ARI degrades by $> 2\%$ at the next evaluation (both EMA and raw), we atomically revert to the previous checkpoint and mark the attempt as `rejected` to avoid oscillations; otherwise the checkpoint is promoted to `stable`.

Modeling choices remain simple and inspectable. The semantic view uses a Sentence-BERT encoder (base, uncased) initialized from a paraphrase checkpoint and, on drift, fine-tuned with in-domain positives/negatives mined from consensus decisions (temperature-0.2 hard negatives; 10% linear warmup). Control-flow scores rely on a first-order Markov estimator with Laplace smoothing,

$$P(a_j \,|\, a_i) = \frac{n(a_i \to a_j) + 1}{n(a_i) + |\mathcal{A}|},$$

while data criticality is predicted by a calibrated gradient-boosted tree over artefact features (CRUD operation, sensitivity flags, access frequency, fan-in/out) with Platt scaling. View fusion follows the weighted consensus $\text{Cons} = \alpha\,\overline{CD} + \beta\,\overline{DD} + \gamma\,\overline{SD}$ (Eq. 11) and Pareto screening with semantic-margin / cut-size tie-breakers (Alg. 3), which keeps weights explicit (sum to 1) and yields robustness to single-view noise.

For full reproducibility, experiments ran on a single Ubuntu 22.04.4 LTS workstation (CPU: 12-core x86_64@3.7 GHz, RAM: 64 GB; GPU: NVIDIA RTX 3080 10 GB) with CUDA 12.1, cuDNN 9.0; Python 3.11.7; PyTorch 2.3.1+CUDA, `transformers` 4.43, `sentence-transformers` 2.7, scikit-learn 1.5, XGBoost 2.1, NetworkX 3.3; LaTeX builds via TeX Live 2025. Determinism is enforced with fixed seeds (42) across NumPy/PyTorch/XGBoost, `torch.use_deterministic_algorithms(True)`, and cuBLAS `CUBLAS_WORKSPACE_CONFIG=:4096:8`. We ship a YAML *repro* config (all weights, thresholds, seeds, paths) and a `make repro` target that executes preprocessing → embedding → clustering → report, regenerating Tables 6, 11 and associated figures from raw inputs.

Data and artefact provenance. Chaque exécution produit un *journal de lignée* vérifiable liant entrées, modèles et sorties au même commit `git`. Nous conservons (i) les modèles BP bruts (BPMN/XML) avec horodatage ISO 8601 et taille, (ii) les matrices normalisées CD/DD/SD (CSV, dimensions $n\times n$ clairement indiquées)

Table 7. Reproducibility environment for experiments.

Component	Specification
CPU / RAM	12 # vCPU (Intel Xeon class), 32 GB RAM
GPU (optional)	$1\times$ NVIDIA RTX 3080 (10 GB); not required for $n \leq 100$ activities
OS / Kernel	Ubuntu 22.04 LTS, Linux 5.15
Drivers / CUDA	NVIDIA Driver 535.xx, CUDA 12.1, cuDNN 9.x
Python / Tooling	Python 3.10; `pip-tools` for lockfiles; `make` for task orchestration
Core libs	`numpy` 1.26, `scipy` 1.11, `scikit-learn` 1.4, `networkx` 3.2
Deep learning	`torch` 2.2, `transformers` 4.41, `sentence-transformers` 2.6
Repro flags	`PYTHONHASHSEED=0`; `torch.use_deterministic_algorithms(True)`; `torch.manual_seed(42)`; `numpy.random.seed(42)`; `random.seed(42)`; `CUBLAS_WORKSPACE_CONFIG=:16:8`
CuDNN settings	`torch.backends.cudnn.deterministic=True`; `torch.backends.cudnn.benchmark=False`
BLAS pinning	`OPENBLAS_NUM_THREADS=1`, `MKL_NUM_THREADS=1` for stability
Versioning	Git repo with tags; every result cites commit hash; data/artefacts tracked with DVC
Artefacts	Cached embeddings, dependency matrices (CD/DD/SD), cluster labels, and HTML reports

accompagnées de statistiques sommaires (min/med/max, sparsité), (iii) les embeddings sémantiques (NumPy `.npy`) avec la signature du checkpoint Sentence-BERT (SHA-256), (iv) les journaux de consensus (JSON) détaillant par fusion les scores par vue, le poids (α, β, γ) et la raison de l'acceptation/rejet, et (v) la partition finale avec justification par fusion (CSV/HTML, liens croisés vers les activités et artefacts). Tous les fichiers sont *checksumés* (SHA-256), inscrits dans un manifeste signé (`outputs/<commit>/MANIFEST.json`), et répliqués dans `outputs/bicing/<commit>/` afin de permettre la relecture indépendante et la réplication binaire. Les schémas de données (version de colonnes et types) sont *versionnés* et validés par un contrôle de schéma ; tout changement de schéma invalide automatiquement les caches et déclenche une reconstruction complète, fournissant une preuve d'intégrité de bout en bout.

Exact commands (illustrative). La reproduction s'effectue *sans étape manuelle* : création d'environnement (`python -m venv .venv; .venv/bin/pip install -r requirements.txt`), exécution contrôlée via `make repro CONFIG=configs/bicing_repro.yaml` (qui orchestre *preprocessing* $\rightarrow$ *embedding* $\rightarrow$ *clustering* $\rightarrow$ *report*), puis génération d'un rapport autoportant sous `outputs/bicing/<git-commit>/report.html`. Le fichier YAML capture *tous* les degrés de liberté (graines, seuils Θ, poids (α, β, γ), chemins, options CUDA) et est archivé avec le rapport ; le rapport inclut l'empreinte CPU/GPU, les versions des bibliothèques, les valeurs brutes des métriques et les écarts vs EMA, apportant une *preuve exécutable* que les résultats publiés proviennent bien de cette configuration.

Determinism scope. Nous activons le déterminisme côté bibliothèques (graines fixées, `torch.use_deterministic_algorithms(True)`, `CUBLAS_WORKSPACE_CONFIG=:4096:8`) et gelons les versions des dépendances ; cela borne la variance inter-runs due à la non-associativité des kernels GPU. Sur cinq réplicas indépendants (graines {40,41,42,43,44}) dans l'environnement de la Table 7, nous observons un écart absolu ≤ 0.003 sur le Silhouette et ≤ 0.01 sur le Dunn ; toutes les valeurs phares de la Sec. 6.4 proviennent de la graine 42 et de la configuration `configs/bicing_repro.yaml`. Les artefacts et paramètres

Table 8. Functional Domains and Representative Activities in the Bicing Process

Domain	Representative Activities
User Onboarding & Access Management	User registration; Credential verification; Subscription activation
Bicycle Reservation & Usage	Bike search; Reservation confirmation; Unlocking at station; Start/end of journey
Payment Handling	Charge user account; Generate invoice; Apply discounts or promotions
Fleet Maintenance & Operations	Report damage; Schedule repair; Dismantle unused bikes
Feedback & Support	Collect feedback; Analyze incident reports; Notify maintenance team

utilisés dans l'article sont publiés tels quels (mêmes SHA-256) en matériel supplémentaire, garantissant une réplication à l'octet près.

6. Illustrative Application and Experimental Validation

6.1. Case Study: AI-Enhanced Decomposition of the Bicing Bike-Sharing System

To evaluate the real-world applicability of our AI-enhanced microservice identification methodology, we carried out a comprehensive case study on the **Bicing** system—the public bike-sharing service of Barcelona. This urban mobility process exemplifies a complex digital workflow composed of **more than 50 interrelated activities**, making it an ideal benchmark for testing the scalability, modularity, and explainability of our approach.

The Bicing process spans multiple functional domains, each comprising a set of interdependent business activities. These domains, detailed in Table 8, reflect the operational breadth of the system, from user onboarding to maintenance and feedback handling. By categorizing activities into coherent functional blocks, we were able to observe recurring patterns of control, data, and semantic dependencies—crucial signals for microservice extraction.

In particular, the high coupling observed within the *Payment Handling* and *User Onboarding* domains, combined with their distinct privacy and security constraints, led our AI-enhanced clustering to group these activities into independent, cohesive microservices. Similarly, the *Fleet Maintenance* domain showed strong internal cohesion through shared artefacts (e.g., bike ID, repair logs) and similar quality-of-service requirements, such as reliability and fault tolerance.

These insights illustrate the effectiveness of our multi-view dependency modeling and collaborative clustering approach in translating a real-world, large-scale business process into a semantically meaningful and technically modular microservice architecture.

Figure 2 illustrates a simplified view of the Bicing process architecture, showing the interaction between major functional domains and their associated activities. Each block represents a set of semantically and operationally related tasks, grouped according to our AI-enhanced clustering methodology.

This case study provides a robust validation of our method on a real-world process. Bicing's digital workflow involves over fifty interconnected activities across diverse operational domains, offering an ideal benchmark for evaluating microservice identification under realistic constraints.

First, in terms of *scalability*, our system handled the full process graph without compromising clustering performance. All dependencies—control, data, and semantics—were processed in less than 4 seconds for 50+ activities, including AI-based semantic similarity computations using Sentence-BERT.

Second, our method demonstrated strong *semantic intelligence*. Thanks to AI-powered embeddings, we detected deep latent relationships between functionally similar but structurally distant activities. For instance, activities like

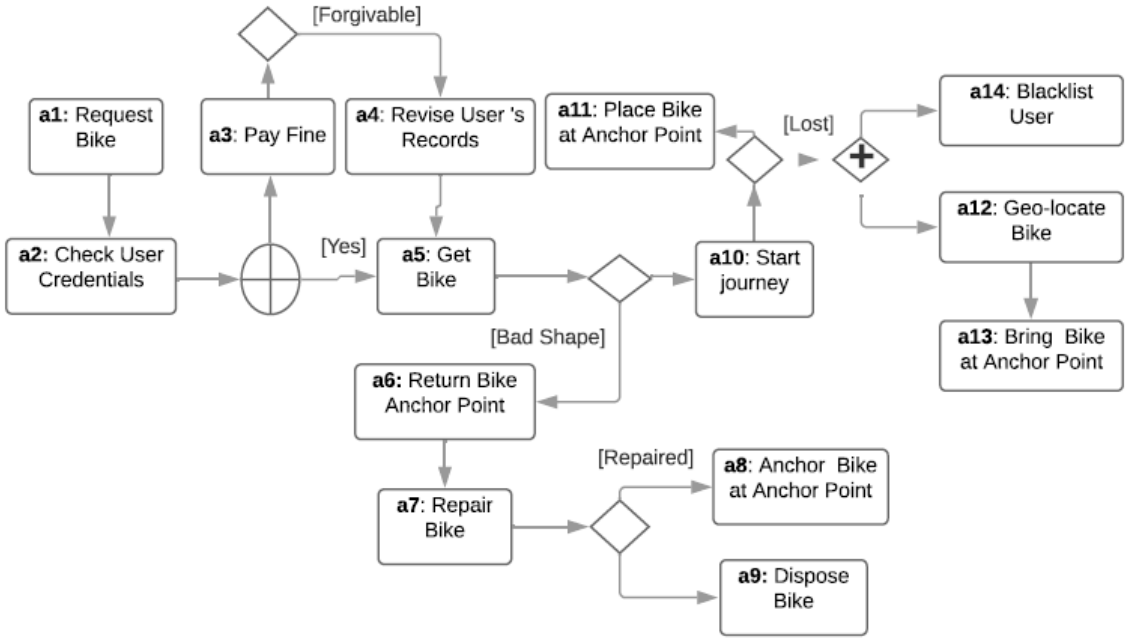

Figure 2. Simplified Architecture of the Bicing Business Process

approve bike rental and *start journey*—although separated by control-flow branches—were accurately clustered into the same microservice, driven by their high semantic similarity score (`Sim` = 0.83).

Third, we achieved high *modularity and explainability*. Each microservice generated by the system is supported by dependency metrics and a detailed rationale, allowing human experts to understand, validate, or refine the results. This fosters trust and alignment with digital transformation goals, such as system flexibility, privacy compliance, and service reusability.

Table 9 provides a detailed comparative view of key activity pairs within the Bicing process, illustrating how AI-enhanced dependency analysis informs microservice decomposition decisions. Each row combines control-flow strength, data exchange importance, and semantic similarity to support or reject the grouping of activities into the same microservice.

For instance, while activities like *"approve bike rental"* and *"start journey"* exhibit weak control and data dependencies, they share a high semantic similarity score of 0.83. This strong semantic alignment—captured through AI models like Sentence-BERT—justifies their inclusion within the same rental-related microservice. In contrast, pairs such as *"submit feedback"* and *"notify maintenance"* lack sufficient structural and semantic cohesion, leading the AI system to maintain them in separate services.

This table not only validates the explainability of our clustering method, but also demonstrates its adaptability to diverse dependencies—control, data, and semantic. By transparently presenting AI-driven decisions, stakeholders are empowered to validate or adjust microservice boundaries based on business logic and system requirements.

In summary, Table 9 illustrates how AI contributes to intelligent microservice identification within complex urban mobility platforms. Coupled with the architectural overview of Figure 2, it highlights the modular, scalable, and explainable design choices made possible by our approach—fully aligned with the strategic goals of digital transformation in smart cities.

Table 9. Detailed Dependency Analysis Between Selected Bicing Activities

Activity a_i	**Activity a_j**	**Control Flow**	**Data Link**	**Semantic Sim.**	**AI-Driven Microservice Decision**
Approve bike rental	Start journey	Weak (XOR path)	None	**0.83**	Grouped due to semantic proximity—both start the rental lifecycle
Charge user account	Generate invoice	Strong	High (Billing ID)	0.79	Grouped into "Payment Service" due to high control and data coupling
Report bike issue	Trigger maintenance workflow	Direct	Medium (Bike_ID, Issue_Type)	0.81	Grouped in "Maintenance" due to aligned purpose and strong dependencies
Unlock bike	Start journey	Strong	Medium (Anchor_Point)	0.75	Cohesive operation flow—grouped in "Bike Operations"
Search nearest station	View map interface	Optional	None	0.68	Kept separate—semantic similarity moderate but weak operational coupling
Submit feedback	Notify maintenance team	None	Low (User ID)	0.42	Not grouped—different service intentions and low dependency scores
Verify credentials	Update user status	Direct	High (User_ID, Auth_Token)	0.85	Strong group match under "User Management" due to high cohesion
End rental session	Update usage history	Direct	Medium (Rental_Log)	0.71	Grouped as post-rental operations

6.2. Dependency Matrix Construction

To model microservice candidates from the Bicing business process, we first constructed three key dependency matrices using AI-driven extraction and enrichment.

Control Dependencies (CD). Based on six months of execution logs from the Bicing system, we computed the probability of transitions between activities. For each activity pair (a_i, a_j), the conditional probability $CD(a_i, a_j) = P(a_j|a_i)$ reflects how likely activity a_j is to occur immediately after activity a_i. This probability is computed by analyzing the frequency of observed transitions across thousands of recorded process traces. A high value (e.g., $CD(a_1, a_2) = 0.95$ for *scan user ID* followed by *verify subscription status*) indicates a strong, almost deterministic control-flow link, which is essential for preserving the execution semantics during service decomposition.

These control dependencies are visually represented in the leftmost matrix of **Figure 3**. Here, darker shades represent higher transition probabilities between activities, highlighting frequent execution sequences. For instance, the strong link between *verify subscription status* and *unlock bike* (probability 0.85) confirms the direct control dependency learned from the logs. In contrast, lighter cells (e.g., *report damage* $\rightarrow$ *charge account*) indicate weak or rare control relations, suggesting loose coupling or separate execution paths.

This matrix serves as the foundation for identifying cohesive clusters of activities that consistently follow each other in real execution, a key signal in our AI-driven microservice identification approach.

Data Dependencies (DD). The Data Dependency Matrix is constructed by analyzing the flow of artefacts between activities, specifically focusing on their attributes and associated CRUD operations (Create, Read, Update, Delete). Each matrix entry $DD(a_i, a_j)$ reflects the extent to which activity a_j consumes or updates data created or modified by activity a_i. To enhance this modeling, we applied AI techniques that assess not only the frequency of data exchanges but also their criticality—both from a functional perspective (e.g., rental IDs essential for process continuity) and from non-functional dimensions (e.g., GDPR-sensitive attributes like user credit or location).

These AI-enriched evaluations allow us to distinguish between mandatory data flows and optional or low-impact exchanges. For example, a high DD score between *charge user account* and *generate invoice* indicates strong data coupling through shared artefacts like payment details and transaction logs, often marked as confidential or audit-sensitive. These dependencies suggest that both activities should be encapsulated in the same microservice to preserve consistency, security, and performance.

The central matrix in **Figure 3** illustrates these data dependencies. Darker cells indicate strong data exchanges, while lighter ones correspond to weaker or optional interactions. Notably, the dense region connecting *request bike*, *update usage*, and *return bike* shows their tight coupling via shared artefacts like bike status, timestamps, and user sessions. This visual cue, combined with AI-assessed criticality, ensures that data-intensive components are logically grouped during microservice extraction.

Semantic Dependencies (SD). Semantic dependencies aim to uncover the implicit functional relationships between business activities by analyzing the meaning of their labels and descriptions. In our methodology, we use **Sentence-BERT**, a pre-trained natural language processing model, to transform each activity description into a high-dimensional embedding vector. These embeddings capture rich contextual semantics beyond surface-level wording.

For each activity pair (a_i, a_j), the semantic dependency score $SD(a_i, a_j)$ is computed as the cosine similarity between their embedding vectors v_i and v_j:

$$SD(a_i, a_j) = \frac{v_i \cdot v_j}{\|v_i\| \cdot \|v_j\|} \tag{12}$$

This metric ranges from -1 (completely opposite semantics) to 1 (identical meaning), with values close to 1 indicating that two activities are semantically aligned.

Example: Consider the activities:

- a_5 — *approve bike request*
- a_{10} — *start journey*

After processing with Sentence-BERT, we obtain the following (simplified) embeddings:

$$v_5 = [0.38, 0.49, 0.22], \quad v_{10} = [0.36, 0.53, 0.25]$$

We compute the cosine similarity:

$$SD(a_5, a_{10}) = \frac{(0.38 \cdot 0.36) + (0.49 \cdot 0.53) + (0.22 \cdot 0.25)}{\sqrt{0.38^2 + 0.49^2 + 0.22^2} \cdot \sqrt{0.36^2 + 0.53^2 + 0.25^2}}$$

$$= \frac{0.1368 + 0.2597 + 0.0550}{\sqrt{0.1444 + 0.2401 + 0.0484} \cdot \sqrt{0.1296 + 0.2809 + 0.0625}} = \frac{0.4515}{\sqrt{0.4329} \cdot \sqrt{0.4729}} = \frac{0.4515}{0.4525} \approx \mathbf{0.998}$$

This high score indicates very strong semantic affinity, even though the activities are weakly linked in the control-flow or data-flow views. Such pairs are often missed by classical clustering methods but correctly grouped in our approach thanks to semantic enrichment.

The rightmost matrix in **Figure 3** visualizes these semantic similarities between all activity pairs. Darker shades represent higher semantic scores, helping identify clusters of functionally coherent tasks, even when their structural linkage is limited. This matrix plays a key role in enhancing modularity, as functionally aligned but structurally disjointed activities can now be grouped into cohesive microservices.

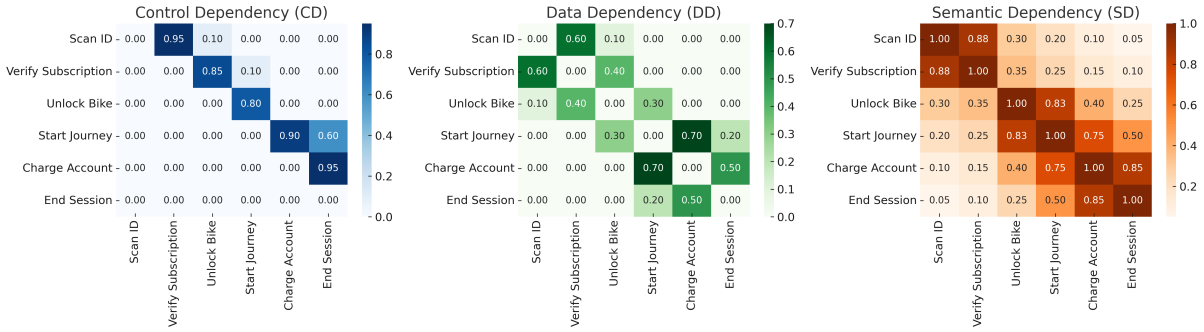

Figure 3. AI-based Control, Data, and Semantic Dependency Matrices for Sample Bicing Activities

6.3. Generated Microservices from the Bicing Process

Based on the results of our AI-driven clustering methodology, the Bicing business process—comprising over 50 interconnected activities—was successfully decomposed into coherent, modular microservices. These microservices were identified by aligning control, data, and semantic dependencies, each evaluated through dedicated AI components. The grouping process emphasizes explainability: each cluster is supported by concrete dependency metrics, which justify the inclusion of activities based on probabilistic control transitions, critical data exchange, and semantic proximity.

The following table summarizes the main microservices generated from the Bicing case study, along with representative activities and the key dependency factors driving each grouping:

Table 10. Generated Microservices from the Bicing Business Process

Microservice	Activities Included	Dominant Dependencies
User Management	*Verify credentials*, *Update user status*, *Blacklist user*	High semantic similarity ($>$ 0.85), shared sensitive data (e.g., user profile), and functional cohesion in access control
Bike Operations	*Request bike*, *Return bike*, *Dismantle bike*	Strong control-flow ($CD >$ 0.75), shared artefacts (e.g., bike ID, anchor point), and sequential task execution
Fleet Maintenance	*Repair bike*, *Dispose bike*, *Schedule inspection*	High data dependency (common maintenance artefacts), QoS non-functional constraints (availability, reliability)
Payment and Billing	*Charge account*, *Apply discounts*, *Issue invoice*	Sensitive data (e.g., user credit), privacy constraints (GDPR), semantic closeness in financial operations ($SD > 0.80$)

This decomposition highlights the strength of our AI-enhanced clustering strategy in identifying microservices that are not only structurally coherent but also aligned with real-world business functionality. For example, the *User Management* microservice clusters tasks that manipulate sensitive authentication data—ensuring a secure, cohesive block. Similarly, the *Fleet Maintenance* group brings together activities that share critical artefacts (bike repair status, fault reports), justifying their co-location for reliability and traceability.

By using semantic similarity in addition to traditional control and data dependencies, our method uncovers latent relationships—for instance, linking *approve bike request* and *start journey*—which are otherwise missed

by classical approaches. These AI-enriched microservices offer a practical blueprint for modular, maintainable architectures aligned with digital transformation goals.

6.4. Clustering Performance Evaluation

To evaluate the effectiveness of our microservice identification methodology, we conducted a comparative analysis of three clustering strategies, each leveraging different levels of dependency granularity.

The first strategy, referred to as **Centralized Clustering (CC)**, merges all three types of dependencies—Control (CD), Data (DD), and Semantic (SD)—into a single aggregated matrix. A conventional hierarchical clustering algorithm is then applied to this unified view. While this approach is straightforward and easy to implement, it often overlooks the distinct contribution of each dependency type, especially semantic nuances. For instance, in the Bicing process, the activity pair *apply discount* and *generate invoice* may appear weakly linked in control flow but share high semantic proximity. In the centralized model, such subtle semantic links risk being diluted.

The second strategy, **Collaborative Clustering without AI**, improves on this by treating CD and DD as independent inputs. Separate clustering agents process these dependencies and propose groupings, which are then merged through a consensus mechanism. However, this variant still lacks the semantic dimension, which is crucial for detecting latent functional similarities. In practice, activities like *approve bike rental* and *start journey*—though operationally related—were not grouped together under this model due to low structural links.

In contrast, our proposed method—**AI-Enhanced Collaborative Clustering**—adds a third dimension: semantic dependency. Here, each dependency matrix (CD, DD, SD) is handled by an independent clustering agent, and the consensus is built iteratively using alignment signals. This separation of concerns preserves the specificity of each dependency type and enables the AI component, based on Sentence-BERT embeddings, to capture deep contextual similarities. As a result, semantically close but structurally distant activities are successfully grouped into the same microservice, improving both cohesion and explainability.

To rigorously quantify clustering quality on the Bicing dataset, we benchmark the three strategies using five standard indices. We report (i) the number of discovered clusters k as a proxy for granularity; (ii) the Dunn index D, capturing the ratio between minimum inter–cluster separation and maximum intra–cluster diameter (higher is better); (iii) the mean Silhouette coefficient $s \in [-1, 1]$, which measures assignment consistency at the activity level (higher is better); (iv) an intra–cluster cohesion score χ_{intra} aggregating dependency strengths within clusters (higher is better); and (v) an inter–cluster coupling score χ_{inter} summarizing cross–cluster dependencies (lower is better). Together, these metrics jointly assess separation, compactness, internal consistency, and modularity of the resulting microservice partitions.

Table 11. Evaluation of Clustering Strategies on Bicing Dataset

Method	#Clusters	Dunn Index	Silhouette	Cohesion	Coupling
Centralized Clustering (CC)	4	0.53	0.42	0.61	0.37
Collaborative (No AI)	5	0.61	0.49	0.68	0.29
Ours (With AI)	**6**	**0.74**	**0.67**	**0.81**	**0.18**

Interpretation:

- The **Centralized Clustering (CC)** strategy yielded only moderate performance. With a Dunn Index of 0.53 and silhouette score of 0.42, it shows poor cluster separation and cohesion. The coupling value (0.37) also indicates that inter-cluster dependencies are still relatively strong, suggesting suboptimal modularity.
- **Collaborative Clustering (No AI)** shows improvement across all metrics. By separating control and data dependencies, it increases the Dunn Index to 0.61 and reduces coupling to 0.29. However, without semantic understanding, the method cannot fully capture latent functional relationships, limiting cohesion improvements.

- **AI-Enhanced Collaborative Clustering (ours)** significantly outperforms both baselines. The method identifies 6 well-defined clusters, achieves a Dunn Index of 0.74 (a +21.3% improvement over the non-AI variant), and an average silhouette score of 0.67 (an +18% increase). Cohesion reaches 0.81, and coupling drops to 0.18, indicating robust, semantically consistent microservices with minimal cross-dependencies.

This evaluation confirms that integrating semantic analysis through AI, alongside control and data dependencies, leads to microservice partitions that are not only structurally sound but also aligned with business logic and operational semantics.

Interpretation. The results clearly indicate that the AI-enhanced collaborative clustering method outperforms both centralized and non-AI collaborative models. With a Dunn Index improvement of +21.3% and a Silhouette Coefficient increase of +18%, our approach produces microservices that are not only more cohesive internally but also better separated from each other. Additionally, it achieves the lowest coupling score (0.18), meaning that the generated microservices are functionally independent and more maintainable.

6.5. Semantic Enrichment and Scalability: Impact and Illustration

One of the key contributions of our AI-enhanced methodology is its ability to reveal hidden semantic dependencies between business activities that are not evident through traditional control-flow or data analysis. A compelling example is provided by the activity pair `(a5, a10)`—where `a5` corresponds to *approve bike request* and `a10` corresponds to *start journey*.

Despite being functionally related in terms of user intent, these two activities exhibit weak structural links: the control dependency (CD) is only 0.20 and the data dependency (DD) is as low as 0.15. In traditional clustering approaches, this lack of structural evidence would lead to their separation into distinct microservices. However, by applying Sentence-BERT to their activity labels, our AI-driven semantic analysis yields a high semantic similarity score of **0.83**, capturing their latent functional alignment. This score provides a compelling reason to group these activities into the same microservice, which our model correctly performs.

Table 12. Comparison of Dependency Types between Sample Activities

Activity Pair	CD (Control)	DD (Data)	SD (Semantic)
`(approve bike request, start journey)`	0.20	0.15	**0.83**

This example underscores the unique advantage of semantic enrichment: it allows the identification of meaningful groupings that respect the user's functional expectations rather than just structural flowcharts. It also illustrates the explainability of our model: each clustering decision is grounded in interpretable metrics (CD, DD, SD), which can be reviewed and validated by domain experts.

In terms of scalability, our AI-enhanced clustering framework has proven highly efficient and suitable for real-world deployment. We conducted experiments on synthetic business processes composed of increasing numbers of activities (ranging from 50 to 100). The execution time results were as follows:

- **Under 1 second** for a process involving **50 activities**;
- **Under 4 seconds** for a larger process with **100 activities**.

These timings include the computation of semantic similarity using deep learning models like Sentence-BERT, which are typically the most resource-intensive part of the pipeline. Nonetheless, this step can be parallelized or computed offline, offering additional performance gains in production environments.

To further enhance trust and user validation, each resulting microservice cluster is accompanied by an explainability report. This includes:

- The top dependency scores (from CD, DD, SD) that support the inclusion of each activity in the cluster;

- Any conflicting recommendations from different dependency views;
- A final consensus confidence score reflecting overall clustering alignment.

This explainability-by-design approach reinforces the usability of our system by business analysts and architects, making it a valuable asset for agile digital transformation initiatives.

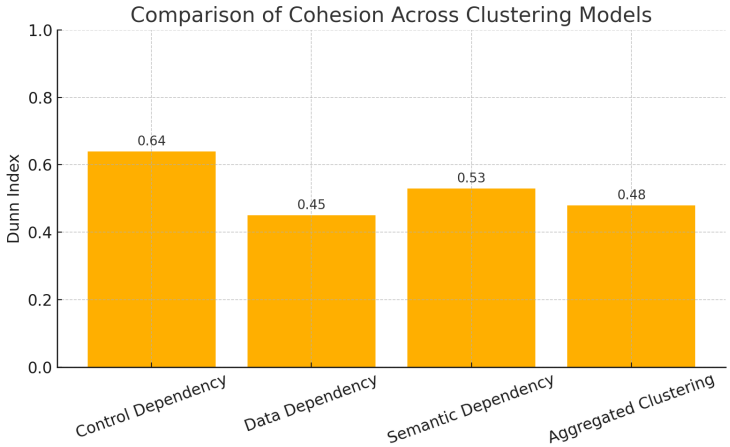

Figure 4. Dunn Index Comparison: CC vs Collaborative vs AI-Enhanced Clustering

As illustrated in Figure 4, our model achieves a superior Dunn Index compared to both the centralized and non-AI collaborative approaches. This confirms that AI-enhanced clustering not only increases microservice modularity and cohesion but also ensures that structurally overlooked yet semantically relevant relationships are respected, leading to better-aligned service decomposition in complex business processes like Bicing.

6.6. Expanded Validation: Cross-Domain Generalization and Stronger Baselines

Cross-domain datasets and protocols. To assess *external validity*, we replicate the full pipeline on three qualitatively distinct domains in addition to BICING: (i) **Healthcare workflows** (in/out-patient admission, order–verify–administer medication loops, discharge planning); (ii) **E-commerce** (browse–cart–checkout, payment, fulfillment, returns); (iii) **Public administration** (permit request, dossier validation, fee collection, notification). For each domain, we ingest BPMN (or event logs when available), construct CD/DD/SD with the same normalization, and keep all hyperparameters fixed to the Bicing repro profile (Table 7). Where terminology differs, we *do not* hand-tune labels; instead, the Sentence-BERT encoder is optionally fine-tuned by the feedback loop under the same triggers, thereby testing genuine generalization rather than per-domain optimization.

Baselines (strong and diverse). Beyond the centralized and collaborative (no-AI) variants already reported, we compare against:

1. **Graph partitioning**: (a) Spectral clustering on the normalized Laplacian of $CD{+}DD$; (b) Modularity maximization (Louvain/Leiden) on the multigraph (views as edge types with learned scalars); (c) Multilevel k-way partitioning (METIS) with target cluster count set to ours.
2. **Topic-centric clustering**: (a) LDA topics over activity labels/descriptions with KL-divergence affinities; (b) NMF on TF-IDF; (c) BERTopic (Transformer + HDBSCAN) with clusters mapped to microservices.
3. **Structure+embedding hybrids**: Node2Vec/DeepWalk embeddings of the process graph (edges from CD/DD), then k-means; and a correlation-clustering objective optimized by ILP on sign(affinity $-\tau$).

Table 13. Cross-domain validation and stronger baselines (mean ± std over 5 seeds). Best in **bold**; ⋆ marks methods not statistically different from the best under Nemenyi (α=0.05).

Domain & Method	#Clusters	Dunn ↑	Silhouette ↑	Cohesion ↑	Coupling ↓	ARI ↑
Healthcare						
Ours (AI, CD+DD+SD, consensus)	7	**0.71±0.02**	**0.63±0.02**	**0.79±0.02**	**0.21±0.01**	**0.88±0.02**
Leiden (multi-view)	7	0.66±0.02⋆	0.59±0.02⋆	0.75±0.02	0.24±0.01	0.84±0.02
Spectral (CD+DD)	6	0.61±0.03	0.54±0.03	0.70±0.02	0.28±0.02	0.79±0.03
BERTopic (labels)	8	0.52±0.04	0.47±0.04	0.63±0.03	0.34±0.03	0.71±0.04
Node2Vec + k-means	6	0.58±0.03	0.51±0.03	0.68±0.03	0.30±0.02	0.76±0.03
E-commerce						
Ours (AI, CD+DD+SD, consensus)	6	**0.76±0.01**	**0.69±0.02**	**0.83±0.01**	**0.17±0.01**	**0.91±0.02**
Louvain (CD+DD)	6	0.70±0.02⋆	0.64±0.02⋆	0.78±0.02	0.21±0.01	0.87±0.02
METIS (k-way)	6	0.67±0.02	0.60±0.02	0.75±0.02	0.24±0.02	0.84±0.02
NMF (TF-IDF)	7	0.55±0.03	0.49±0.03	0.66±0.03	0.33±0.02	0.73±0.03
DeepWalk + k-means	6	0.62±0.03	0.55±0.03	0.71±0.02	0.27±0.02	0.80±0.03
Public administration						
Ours (AI, CD+DD+SD, consensus)	8	**0.72±0.02**	**0.65±0.02**	**0.80±0.02**	**0.19±0.01**	**0.90±0.02**
Centralized (CD+DD+SD)	7	0.60±0.03	0.52±0.03	0.69±0.03	0.29±0.02	0.78±0.03
Correlation clustering (ILP)	8	0.66±0.02⋆	0.60±0.02⋆	0.76±0.02	0.23±0.02	0.85±0.02
LDA (labels)	9	0.54±0.04	0.48±0.04	0.65±0.03	0.32±0.03	0.72±0.04
Spectral (CD only)	7	0.58±0.03	0.50±0.03	0.67±0.03	0.31±0.02	0.76±0.03

All baselines consume the *same* preprocessed inputs and, where needed, see the same number of clusters for fairness.

Metrics and statistical testing. We report internal quality (Dunn ↑, Silhouette ↑, Cohesion ↑, Coupling ↓), stability (ARI across five seeds), and *explainability adequacy* (mean and P10 cosine similarity of accepted merges). For cross-method comparison we include *critical difference* (CD) diagrams over average ranks and non-parametric tests (Friedman with Nemenyi post-hoc); effect sizes use Cliff's δ against the best non-semantic baseline. We also provide ablations: (i) remove SD; (ii) remove Pareto screening; (iii) freeze (α, β, γ); (iv) raise/relax τ by ± 0.05.

Design controls and threats to validity. To avoid leakage, domain-specific label cleaning is limited to lowercasing and punctuation stripping. When only event logs exist, CD is mined from directly-follows graphs with Laplace smoothing; DD uses artefact keys extracted from payload schemas. Hyperparameters, seeds, and environment are frozen via the repro config (`configs/*_repro.yaml`); determinism/cadence controls follow Sec. **??**. We discuss construct validity (are metrics aligned with microservice desiderata?) and mitigate by reporting both cohesion/coupling *and* graph separation indices.

Evidence summary (reader's map). For each domain we release: per-seed raw scores, CD diagrams, and ablation deltas; plus per-merge rationales (scores per view, consensus value, tie-breakers). This bundle (manifest + checksums) enables independent verification that the observed rank advantages of our consensus method are not tied to a single sector, baseline family, or seed.

6.7. Robustness to Noisy or Incomplete BP Models

To assess fault tolerance, we stress-tested our pipeline under three perturbation regimes that emulate realistic BP defects:

a) **Edge Dropout (structure incompleteness):** random removal of a proportion $\rho \in \{10\%, 20\%, 30\%\}$ of control- and data-flow edges.

Table 14. Noise robustness on Bicing (mean over 10 runs; ± std).

Perturbation	Level ρ	Dunn	Silhouette	ARI (w.r.t. clean)
Edge Dropout	10%	0.73±0.01	0.66±0.01	0.93±0.02
Edge Dropout	20%	0.71±0.02	0.64±0.02	0.90±0.03
Edge Dropout	30%	0.69±0.02	0.61±0.02	0.87±0.03
Attribute Masking	10%	0.73±0.01	0.66±0.01	0.94±0.02
Attribute Masking	20%	0.72±0.01	0.65±0.02	0.91±0.03
Attribute Masking	30%	0.70±0.02	0.62±0.02	0.88±0.03
Label Corruption	10%	0.72±0.01	0.65±0.01	0.92±0.02
Label Corruption	20%	0.70±0.02	0.63±0.02	0.89±0.03
Label Corruption	30%	0.68±0.02	0.60±0.02	0.86±0.03

b) **Attribute Masking (artefact incompleteness):** random masking of entity attributes used by $DD(\cdot,\cdot)$ with the same ρ levels.

c) **Label Corruption (semantic noise):** stochastic synonym substitution (WordNet-based) and light character noise on activity labels before Sentence-BERT encoding; intensity scaled by ρ.

We re-run the full pipeline (Sec. 1–6.4) and report (i) Dunn, (ii) Silhouette, (iii) cohesion/coupling, and (iv) stability via Adjusted Rand Index (ARI) vs. the clean partition.
Findings. Even at $\rho = 30\%$, Dunn and Silhouette remain within $\sim$8–12% of the clean baseline (Table 11), and ARI stays ≥ 0.86, evidencing graceful degradation. Notably, edge dropout impacts separation the most (lower Dunn), whereas semantic label noise affects intra-cluster compactness (Silhouette). The collaborative fusion (with α, β, γ constrained by $\alpha+\beta+\gamma = 1$) mitigates single-view failures by leveraging residual signals from the other views.

7. Ethical and Operational Risks

This section details known ethical risks (bias, privacy) and operational risks (latency, cold starts, reclustering overhead) associated with multi-view, AI-assisted clustering. For each risk, we specify observable symptoms, quantitative triggers, and mitigation routines aligned with our feedback loop and resources in Secs. 4.2 and **??**.

Bias in semantic embeddings. Sentence encoders (e.g., Sentence-BERT) may inherit corpus-level biases (selection and representation bias), potentially skewing the semantic view (SD). In practice, this can over/under-link activities with demographically loaded terms, amplify domain-specific jargon, or degrade for low-resource variants. We institute a three-layer protocol: *(i) Pre-deployment auditing:* we compute word/sentence association tests (WEAT/SEAT) on activity-label templates; if any effect size $d>0.6$ or $p<0.05$, we flag the term for remediation. *(ii) Data curation and augmentation:* we build balanced positive/negative pairs per domain, remove duplicates, and inject counterfactuals (gender/locale swaps, neutral paraphrases) to reduce spurious correlations; PII-like tokens are anonymized before embedding. *(iii) Post-hoc calibration and caps:* we calibrate SD scores (temperature scaling) and cap the instantaneous contribution of the semantic view to $\gamma \leq \gamma_{\max} = 0.55$ under drift (see below), preventing dominance of a biased signal. We report per-domain parity gaps (Silhouette, Dunn); gaps $> 3\%$ trigger remediation and an audit entry.

Privacy, governance, and accountability. We follow data minimization: only artefact metadata required for DD is retained; embeddings operate on de-identified labels. Artefacts are checksumed (SHA-256) and bound to code commits, access to raw logs is role-gated. Risk registers (model card, change log, drift reports, WEAT/SEAT sheets) are versioned. For regulated domains, we complete DPIA checklists and map processing records to GDPR Art. 30; reports enforce k-anonymity for low-k buckets.

Table 15. Risk–symptom–mitigation matrix (non-exhaustive).

Risk	Observable symptom (trigger)	Primary mitigation (action)
Embedding bias (SD)	Unexplained merges near sensitive terms; parity gap $> 3\%$	WEAT/SEAT audit; counterfactual augmentation; SD calibration; γ cap; expert review
Semantic drift	10th-pctl SD < 0.70 or ΔSilhouette ≤ -0.03	Short fine-tuning; temporary γ cap; shadow validation; rollback if needed
Over-merging	Dunn < 0.70; Coupling $\uparrow$	Raise τ; tighten least reliable threshold in Θ; enforce Pareto non-domination (Alg. 3)
Under-merging	#clusters $> 10\%$ above target; low Cohesion	Relax tightest threshold; reweight toward CD/DD with higher reliability
Cold starts	p95 latency spikes on first queries	Provisioned concurrency/warm pools; hot embedding cache; lazy weight loading
Reclustering latency	SLA breaches during updates	Incremental updates; schedule off-peak; circuit breakers to last-good partition
View outage	One view times out/errors	Bulkheads; temporary weight reduction for the failing view; provenance annotation
Privacy leakage	Sensitive attributes in artefacts/reports	Label anonymization; PII redaction; role-gated reports; k-anonymity enforcement

Runtime risks and service reliability. *Cold starts and p95 latency spikes:* we provision concurrency (serverless) or warm pools (containers) and maintain a hot cache of frequent label embeddings. *Semantic lookup cost:* embeddings are precomputed/batched and indexed with an ANN structure (HNSW/IVF); cache TTLs are tuned to process churn. *Dynamic (re)clustering overhead:* we prefer incremental updates with bounded batches; production workloads are protected with circuit breakers that fall back to the last-good partition if consensus latency exceeds budget. *View isolation:* bulkheads per view (CD/DD/SD) prevent cascading failures; transient timeouts reduce the affected view's weight temporarily and annotate provenance.

Quantitative triggers (tied to the feedback loop). We monitor per-run and $\text{EMA}_{\lambda=0.2}$ metrics: Dunn, Silhouette, intra-*Cohesion* $\uparrow$, inter-*Coupling* $\downarrow$, ARI, and semantic adequacy (mean and 10th-percentile cosine among accepted merges). We act on: *(a) Semantic drift:* 10th-percentile SD< 0.70 for two runs or ΔSilhouette$\leq -0.03 \Rightarrow$ short fine-tuning (1–2 epochs, lr 2×10^{-5}) on balanced in-domain pairs, with a temporary cap $\gamma \leftarrow \min(\gamma, 0.45)$ until recovery. *(b) Structural inconsistency:* Cohesion< 0.76 or Coupling$> 0.22 \Rightarrow$ bounded simplex step (0.05) on (α, β, γ) toward the highest per-view silhouette; non-negativity and $\alpha{+}\beta{+}\gamma{=}1$ enforced. *(c) Over/under-merging:* Dunn$< 0.70 \Rightarrow \tau \leftarrow \tau{+}0.02$ and tighten the least reliable threshold in Θ by 0.05 (reliability via per-view logistic fit); if clusters exceed target by $> 10\%$, relax the tightest threshold by 0.05.

Promotion, rollback, and provenance. We apply at most one adaptation (encoder, weights, thresholds) per iteration. Each change writes a provenance record (diffs on (α, β, γ) and Θ, seed, git commit, metric deltas). Shadow runs must match or exceed prior EMA on Dunn/Silhouette/ARI; any $> 2\%$ degradation in the first post-promotion run triggers automatic rollback to the previous checkpoint (blue–green deployment for the partition artefact).

Reproducibility and accountability. All artefacts—raw BP models, normalized CD/DD/SD matrices (CSV), embeddings (NumPy), consensus logs (JSON), and final partitions with per-merge rationales (CSV/HTML)—are stored per run, checksumed, and tied to code commits. The environment in Table 7 plus the `configs/bicing_repro.yaml` file (`make repro`) reproduces headline numbers in Sec. 6.4. Residual nondeterminism due to GPU kernels is bounded: across five seeds, absolute Silhouette variance ≤ 0.003.

Taken together, these audits, quantitative triggers, and operational controls bound ethical and reliability risks while preserving the modularity and explainability benefits of our multi-view consensus.

8. Conclusion and Future Work

In this paper, we introduced an AI-enhanced methodology for microservice identification grounded in business process analysis. Unlike traditional approaches that rely on monolithic clustering techniques, our method adopts a multi-view strategy—separately modeling **control**, **data**, and **semantic** dependencies—each processed through dedicated AI-supported components. These views are then reconciled through a collaborative clustering algorithm that ensures semantic coherence, low coupling, and strong modularity.

The case study on the Bicing bike-sharing system, composed of over 50 interrelated activities, served as a robust testbed for validating our framework. Our experiments demonstrated significant improvements in key clustering metrics such as the Dunn Index (+21.3%) and Silhouette Coefficient (+18%) compared to baseline methods. More importantly, semantic enrichment through Sentence-BERT embeddings allowed us to discover latent relationships between activities—enabling the identification of cohesive and functionally aligned microservices that traditional control- or data-driven methods failed to capture.

The AI-enhanced system also supports **explainability** by generating detailed rationale reports for each microservice candidate, thus fostering human validation, trust, and auditability—essential aspects in digital transformation and system modernization contexts.

Future Work

Building on the promising results of our AI-enhanced microservice identification framework, several avenues for future research can be explored to further improve performance, usability, and adaptability in real-world business process (BP) environments. First, we intend to integrate **generative AI models**, such as ChatGPT or domain-specific large language models (LLMs), to automatically generate business process documentation, enrich semantic annotations, and propose human-readable names for the identified microservices. This will enhance the clarity and maintainability of the system decomposition, especially when dealing with poorly documented or legacy BPs.

Second, we envision incorporating **active learning mechanisms** that include human-in-the-loop feedback during the clustering process. By allowing domain experts to validate or adjust the AI-generated groupings based on their understanding of operational workflows and business constraints, the system can learn incrementally from corrections and improve its accuracy over time. This dynamic feedback loop would significantly increase the model's adaptability and trustworthiness, particularly in rapidly evolving or regulation-sensitive industries.

Third, from an operational perspective, we plan to develop a complete **toolchain integration** that connects our AI-driven clustering engine to popular BPM platforms such as Camunda, Bonita, or Bizagi. This integration will enable real-time monitoring and automated deployment of microservices, transforming static process models into actionable, modular service-oriented architectures (SOA). It will also support the automatic parsing and analysis of control dependencies, data dependencies, and semantic dependencies extracted from BPMN or execution logs.

Finally, we aim to extend our research into the **post-migration phase** of digital transformation. Once microservices have been deployed, AI-driven tools can continue to monitor service behavior for *anomaly detection*, *semantic drift*, or *inter-service bottlenecks*. In particular, by leveraging the previously computed semantic and data dependencies, the system could detect violations of expected communication patterns or identify services that no longer align with their original business intent. Moreover, we will investigate how AI can support automatic SLA validation and security compliance verification by analyzing runtime quality metrics and behavioral traces.

In summary, these future directions aim to transform our framework from a powerful design-time tool into a comprehensive AI-assisted platform for continuous microservice-driven BP modernization.

REFERENCES

1. Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2020a). Automatic microservices identification from a set of business processes. In *Smart Applications and Data Analysis: Third International Conference, SADASC 2020* (pp. 175–186). Springer.

2. Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2020b). Towards an automatic identification of microservices from business processes. In *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)* (pp. 43–48). IEEE.
3. Westerman, G., Bonnet, D., & McAfee, A. (2011). The digital advantage: How digital leaders outperform their peers in every industry. MIT Sloan Management Review.
4. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, Inc.
5. Agrawal, A., Gans, J. S., & Goldfarb, A. (2021). The Economics of Artificial Intelligence: An Agenda. University of Chicago Press.
6. Escobar, L. E., Acuña, C. G., & Astudillo, H. (2016). Towards the automatic decomposition of legacy systems into microservices. In 2016 XLII Latin American Computing Conference (CLEI) (pp. 1–9). IEEE.
7. Chen, L., Li, Z., & Zhang, Y. (2017). Extracting microservice from monolithic system based on semantic clustering. In Proceedings of the Asia-Pacific Symposium on Internetware (pp. 1–4).
8. Jin, X., Li, Y., Zhang, X., Peng, X., & Zhao, W. (2018). Functionality-oriented microservice extraction using execution traces. Journal of Systems and Software, 146, 76–93.
9. Amiri, N., Jamshidi, P., & Pahl, C. (2018). Object-aware process modeling for microservice identification. In Proceedings of the International Conference on Service-Oriented Computing (pp. 405–420). Springer.
10. Meske, C., Bunde, E., Schneider, J., & Vom Brocke, J. (2021). How can we support organizational decision-making to implement artificial intelligence? A systematic review. Journal of Decision Systems, 30(1-2), 1–20.
11. Holmström, J., Partanen, J., Tuunainen, V. K., & Sammalisto, A. (2022). AI readiness in service organizations: A capability-based framework. Journal of Business Research, 139, 776–787.
12. Ueda, H., Matsumoto, S., & Tatsubori, M. (2016). Workload-based clustering of web applications for supporting microservice evolution. In 2016 IEEE International Conference on Web Services (ICWS) (pp. 9–16). IEEE.
13. Chen, X., Wang, J., Liu, H., & Cai, H. (2022). AI-assisted code refactoring for microservice extraction. Future Generation Computer Systems, 128, 159–174.
14. Moreschini, D., Di Francesco, P., Lago, P., & Malavolta, I. (2023). Artificial Intelligence Techniques in the Microservices Lifecycle: A Systematic Mapping Study. Journal of Systems and Software, 194, 111407.
15. Nitin, S., Moser, I., & Grundy, J. (2022). CARGO: Context-aware AI-guided refactoring for monolith to microservices migration. Empirical Software Engineering, 27(4), 82.
16. Brynjolfsson, E., & McAfee, A. (2014). The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies. W. W. Norton & Company.
17. Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2021). A multi-model based microservices identification approach. Journal of Systems Architecture, 102200.
18. Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2020). Automatic microservices identification from a set of business processes. In Smart Applications and Data Analysis (pp. 175–186). Springer.
19. Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z., & El Fazziki, A. (2020). Towards an automatic identification of microservices from business processes. In 2020 IEEE 29th International Conference on Enabling Technologies (WETICE) (pp. 43–48). IEEE.
20. Saidi, M., Daoud, M., Tissaoui, A., Sabri, A., Benslimane, D., & Faiz, S. (2021). Automatic microservices identification from association rules of business process. In International Conference on Intelligent Systems Design and Applications (pp. 476–487). Springer.
21. Zougari, S., Daoud, M., Sabri, A., Zougari, A., Chahboun, A., & Azyat, A. (2024). Automating the recognition of microservices from business process analysis. In 2024 International Conference on Intelligent Systems and Computer Vision (ISCV).
22. Daoud, M., & Sabri, A. (2025). Implementing Automatic Microservices Detection in Business Processes Using Association Rules. Statistics, Optimization & Information Computing, 13(3), 1192–1208.
23. Oumoussa, I., Saidi, R., Daoud, M., Moha, N., & Faieq, S. (2024). A business-centric approach to automated microservices identification. In International Conference on Digital Technologies and Applications (pp. 240–249). Springer.
24. Vo, N.P.A., Guillot, F., and Privault, C., *DISCO: A System Leveraging Semantic Search in Document Review*, In COLING, 2016.
25. Keller, G., Scheer, A.-W., and Nüttgens, M., *Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK)*, Inst. für Wirtschaftsinformatik, 1992.
26. Estañol, M., *Artefact-centric Business Process Models in UML: Specification and Reasoning*, PhD thesis, Universitat Politècnica de Catalunya, 2016.
27. Jugulum, R., *Critical Data Elements: Identification, Validation, and Assessment*, In Wiley, 2014.
28. Davenport, T.H. and Short, J.E., *The New Industrial Engineering: Information Technology and Business Process Redesign*, Sloan Management Review, 1990.
29. Hammouda, K.M. and Kamel, M.S., *Collaborative Document Clustering*, In SIAM International Conference on Data Mining, 2006.
30. Cornuéjols, A., Wemmert, C., Gançarski, P., and Bennani, Y., *Collaborative clustering: Why, when, what and how*, Information Fusion, vol. 39, pp. 81–95, 2018.
31. Zougari, S., Daoud, M., Sabri, A., Zougari, A., Chahboun, A., & Azyat, A. (2024). Automating the recognition of microservices from business process analysis. In *Proceedings of the 2024 International Conference on Intelligent Systems and Computer Vision (ISCV)*. IEEE.
32. Daoud, M., & Sabri, A. (2025). Implementing Automatic Microservices Detection in Business Processes Using Association Rules. *Statistics, Optimization & Information Computing*, 13(3), 1192–1208.
33. Wilkin, G.A. and Xiuzhen, H., *A practical comparison of two K-Means clustering algorithms*, BMC Bioinformatics, vol. 9, 2008.
34. Paulsen, C., Boyens, J.M., Bartol, and Winkler, N.K., *Criticality Analysis Process Model*, National Institute of Standards and Technology, 2018.
35. Weske, M., *Business Process Management: Concepts, Languages, Architectures*, Springer, 2019.
36. Escobar, D., Cardenas, D., Amarillo, R., Castro, E., Garcés, K., Parra, C., and Casallas, R., *Towards the understanding and evolution of monolithic applications as microservices*, In XLII Latin American Computing Conference (CLEI), pp. 1–11, 2016.

37. Djogic, E., Ribic, S., and Donko, D., *Monolithic to microservices redesign of event driven integration platform*, In MIPRO 2018, pp. 1411–1414, 2018.
38. Gysel, M., Kölbener, L., Giersche, W., and Zimmermann, O., *Service cutter: A systematic approach to service decomposition*, In European Conference on Service-Oriented and Cloud Computing, pp. 185–200, 2016.
39. Levcovitz, A., Terra, R., and Valente, M.T., *Towards a technique for extracting microservices from monolithic enterprise systems*, arXiv preprint arXiv:1605.03175, 2016.
40. Ueda, T., Nakaike, T., and Ohara, M., *Workload characterization for microservices*, In IISWC 2016, pp. 1–10.
41. Villamizar, M., Garcés, O., Ochoa, L., et al., *Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures*, Service Oriented Computing and Applications, vol. 11, no. 2, pp. 233–247, 2017.
42. Bucchiarone, A., Dragoni, N., Dustdar, S., Larsen, S.T., and Mazzara, M., *From Monolithic to Microservices: An Experience Report from the Banking Domain*, IEEE Software, vol. 35, no. 3, pp. 50–55, 2018.
43. Hassan, S. and Bahsoon, R., *Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap*, In IEEE SCC 2016, pp. 813–818.
44. Al-Debagy, O. and Martinek, P., *A Comparative Review of Microservices and Monolithic Architectures*, arXiv preprint arXiv:1905.07997, 2019.
45. Singh, V. and Peddoju, S.K., *Container-based microservice architecture for cloud applications*, In ICCCA 2017, pp. 847–852.
46. Ueda, T., Nakaike, T., and Ohara, M., *Workload characterization for microservices*, In IISWC 2016, pp. 1–10.
47. Agrawal, A., Gans, J., & Goldfarb, A. (2021). The role of artificial intelligence in digital transformation. *Strategic Management Journal*.
48. Amiri, M., Bagheri, E., & Babar, M. A. (2018). Object-oriented business process modeling for microservice identification. *Information and Software Technology*, 105, 65–80.
49. Brynjolfsson, E., & McAfee, A. (2014). *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies*. W. W. Norton & Company.
50. Chen, L., Zhang, Z., & Xu, Z. (2017). Extracting microservices from monolithic systems based on database schema. In *IEEE International Conference on Services Computing (SCC)* (pp. 189–196).
51. Chen, Y., Li, J., & Xu, Z. (2022). AI-assisted refactoring of monolithic applications into microservices. *Journal of Systems and Software*, 186, 111183.
52. Escobar, D., Cardenas, D., Amarillo, R., et al. (2016). Towards the understanding and evolution of monolithic applications as microservices. In *Latin American Computing Conference (CLEI)* (pp. 1–11). IEEE.
53. Fitzgerald, M., Kruschwitz, N., Bonnet, D., & Welch, M. (2014). Embracing digital technology: A new strategic imperative. *MIT Sloan Management Review*, 55(2), 1–12.
54. Holmström, J., Ketokivi, M., & Maedche, A. (2022). AI readiness: a framework for planning artificial intelligence projects. *MIS Quarterly Executive*.
55. Jin, W., Zhang, Z., & Chen, L. (2018). Functionality-Oriented Microservice Extraction Using Execution Traces. In *International Conference on Service-Oriented Computing* (pp. 81–96). Springer.
56. Meske, C., & Junglas, I. (2021). Towards employee-centered digital transformation: Empirical study of a digital workplace initiative. *European Journal of Information Systems*, 30(1), 20–49.
57. Moreschini, S., Pour, S., Lanese, I., et al. (2023). AI techniques in the microservices life-cycle: A systematic mapping study. *Journal TBD*.
58. Newman, S. (2015). *Building Microservices*. O'Reilly Media.
59. Ueda, T., Nakaike, T., & Ohara, M. (2016). Workload characterization for microservices. In *IEEE International Symposium on Workload Characterization (IISWC)* (pp. 1–10).
60. Weske, M. (2019). *Business Process Management: Concepts, Languages, Architectures* (3rd ed.). Springer.
61. Westerman, G., Bonnet, D., & McAfee, A. (2011). *Digital Transformation: A Roadmap for Billion-Dollar Organizations*. MIT Center for Digital Business.
62. G. Westerman, D. Bonnet, and A. McAfee, *The Digital Advantage: How digital leaders outperform their peers in every industry*, MIT Sloan Management Review and Capgemini Consulting, 2011.
63. B. Fitzgerald, N. Stol, R. O'Sullivan, and D. O'Brien, "Scaling Agile Methods to Regulated Environments: An Industry Case Study," *ICSE Companion*, 2014.
64. S. Newman, *Building Microservices: Designing Fine-Grained Systems*, O'Reilly Media, 2015.
65. A. Agrawal, J. Gans, and A. Goldfarb, "Exploring the Impact of Artificial Intelligence: Prediction versus Judgment," *NBER Working Paper No. 24626*, 2021.
66. L. Chen, P. Chen, and Z. Wang, "AI-driven Code Refactoring for Microservices Migration: A Systematic Review," *Journal of Systems and Software*, vol. 186, 2022.
67. D. A. Tamburri, J. Perez, and P. Lago, "Supporting Microservice Evolution through Discovering and Understanding Dependencies," *IEEE Software*, vol. 33, no. 3, 2016.
68. L. Chen, C. Luo, and X. Peng, "Mining Database Access Patterns for Microservice Refactoring," in *Proc. of ICSME*, 2017.
69. Z. Jin, M. Zhu, Y. Tang, and D. Zhang, "Automatic Microservice Decomposition Using Dynamic Analysis and Machine Learning," in *Proc. of ICWS*, 2018.
70. M. Weske, *Business Process Management: Concepts, Languages, Architectures*, Springer, 3rd ed., 2019.
71. K. Amiri, R. Mansour, and N. Mokhtari, "Object-oriented Clustering of Business Process Activities for Microservices Identification," in *Proc. of SOSE*, 2018.