Multi-View Spectral Clustering for Monolith-to-Microservices Migration

Gonzalo Rivera-Lazo[1], **Fernando Montoya**[1,2,3], **Hernán Astudillo**[4]

[1]AISA Research Group – Corriente Sur
Valparaíso, Chile

[2]Universidad Técnica Federico Santa María
Santiago, Chile

[3]Fundación Instituto Profesional Duoc UC
Santiago, Chile

[4]ITiSB - Universidad Andrés Bello
Viña del Mar, Chile

gonzalo@corrientesur.com,fernando.montoya@usm.cl,hernan.astudillo@unab.cl

Abstract. *The migration from monolithic architectures to microservices poses significant challenges, particularly in achieving modularity, decoupling and functional coherence. This paper explores a multi-view spectral clustering approach to systematically decompose monoliths by integrating structural dependencies, domain semantics and dynamic behavior into a unified weighted affinity model. The method uncovers optimal service boundaries that balance cohesion, coupling and complexity, generating multiple architectural options tailored to weighted quality attributes. Validation on a real-world application demonstrates how the framework enables flexible, quality-driven decompositions, supporting more informed and adaptive architectural decisions.*

1. Introduction

The migration from monolithic architectures to microservices is a key strategy in modern software engineering, offering benefits like scalability, maintainability and resilience [Richardson 2018]. However, determining optimal service boundaries remains a significant challenge, as it requires balancing multiple quality attributes—such as coupling, cohesion, performance and development costs—based on business priorities. While state-of-the-art techniques like static analysis, graph clustering and machine learning assist in automating microservice decomposition [Gysel et al. 2016, Mazlami et al. 2017, Taibi and Systä 2019], they often rely on single-metric optimization. This narrow focus limits their ability to address the multidimensional and dynamic nature of real-world architectural decision-making.

Microservice migration is not merely a technical refactoring but a strategic transformation that involves balancing multiple trade-offs, as illustrated in Figure 1. Architects must evaluate how different service boundaries impact performance, development complexity and adaptability. For instance, some services may prioritize latency optimization, while others focus on modularity. Existing approaches lack systematic guidance for assessing these trade-offs, leaving architects without structured methods to explore multiple

viable decompositions. Quality attributes play a fundamental role in shaping architectural decisions. Software quality standards define models and attributes that influence decomposition strategies [Lago and van Vliet 2005]. For example, scalability and availability are crucial in microservices due to their impact on system resilience, while adaptability is essential in dynamic environments [Papazoglou and van den Heuvel 2007]. Current methods fail to provide a parametrizable approach over multiples views, forcing architects to manually adjust service boundaries when automated partitions do not align with business goals.

To address this gap, we introduce multi-view spectral clustering as a principled approach for microservice decomposition. Multi-view clustering refers to the process of integrating multiple heterogeneous views—such as structural dependencies, domain semantics and runtime performance—into a unified model that reveals latent grouping structures [Chao et al. 2021, Zhou et al. 2024]. In our approach, these views are encoded into a weighted affinity matrix, allowing architects to prioritize quality attributes dynamically via a tunable weighting system. Spectral clustering is then applied to this multi-view matrix to identify candidate decompositions that are both technically coherent and aligned with business goals. To determine the optimal number of services, we employ an iterative selection process driven by a composite quality function that balances cohesion, coupling and complexity—enabling dynamic control over microservice granularity.

The remainder of this paper is structured as follows: Section 2 reviews existing approaches to microservice decomposition. Section 3 details our multi-view spectral clustering approach. Section 4 presents experimental results and evaluates our method. Finally, Section 5 summarizes our findings and outlines future research directions.

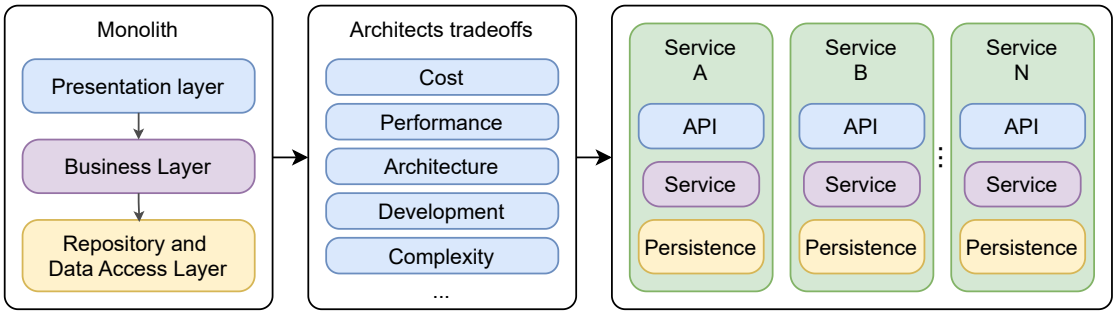

Figure 1. Architectural tradeoffs in the iterative migration of a monolithic application to a microservices-based architecture.

2. Related Work

The migration of monolithic systems to microservices is a well-studied problem with various proposed approaches. This shift was initially driven by industry demands to decouple large, tightly integrated systems into modular architectures, aiming to improve primarily maintainability, enhance scalability and reduce infrastructure costs.

The most related work is a study from [Asseldonk 2021] that evaluates the impact of incorporating multiple viewpoints (static, semantic and dynamic dependencies) on microservices decomposition quality. Results show that combining static and dynamic data

improves modularity, but adding semantic information decreases decomposition quality due to an overemphasis on semantic dependencies.

[Andrade et al. 2023] conducts a comparative analysis of static and dynamic techniques, concluding that neither approach consistently outperforms the other. However, the dynamic data collection process requires significantly more effort.

Other approaches, such as execution trace analysis, infer service partitions based on dynamic behavior [Mazlami et al. 2017]. While this improves accuracy in runtime dependency detection, it overlooks domain-driven constraints that influence maintainability and business alignment. Business process mining techniques [Taibi and Systä 2019] enhance microservice identification by emphasizing functional cohesion, yet they remain limited in incorporating performance metrics and technical dependencies.

Machine learning-based techniques, such as repository mining and historical code evolution analysis, have also been explored [Fritzsch et al. 2019]. [Andrade et al. 2022] compared static and dynamic analysis techniques for identifying candidate microservices. Their study demonstrated that neither approach consistently outperforms the other, though dynamic data collection introduces higher overhead. They utilized a complexity metric to assess the effectiveness of each decomposition strategy.

Given the current state-of-the-art in single-perspective methods, we identified an opportunity to apply a multi-view spectral clustering approach that integrates multiple architectural perspectives: structural, semantic and performance driven into a unified model that outputs a solution related to a combination of quality attributes.

3. Proposal

We propose a Multi-view spectral clustering approach to segment monolithic applications into microservices as shown in Figure 2. By integrating multiple views, our method generates multiple architecture options, enabling architects to dynamically prioritize quality attributes based on business and technical needs.

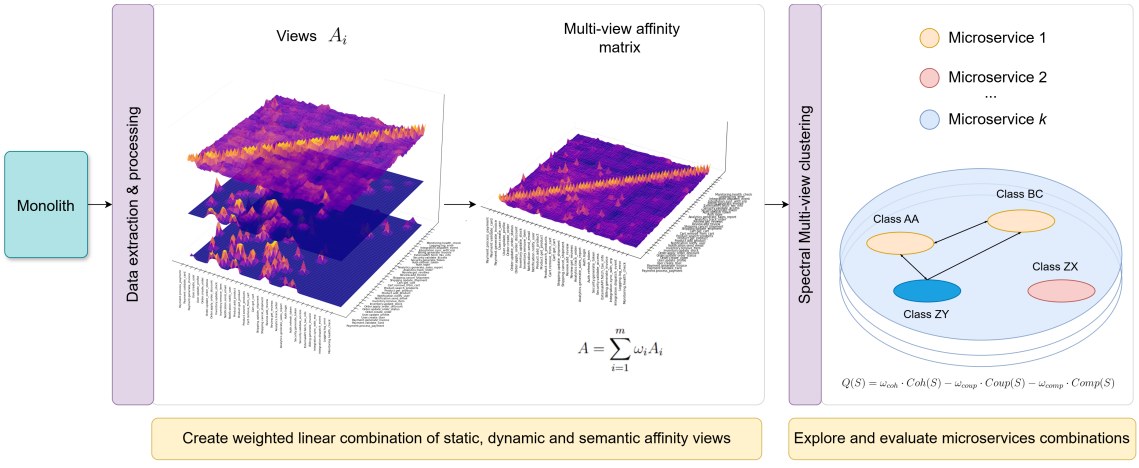

Figure 2. Multi-view spectral clustering pipeline, integrating dependency graphs, dynamic call graphs and semantic data to derive microservices architectures.

3.1. Problem Formulation

Given a monolithic system M composed of n components $C = \{C_1, C_2, ..., C_n\}$, our objective is to partition M into k microservices $S = \{S_1, S_2, ..., S_k\}$ while optimizing multiple quality attributes, such as cohesion, coupling, maintainability and scalability. These attributes are weighted according to a predefined prioritization scheme, allowing the exploration of multiple architectural alternatives.

To achieve this, we first create multiple affinity matrices $A_c, A_d, A_p, \ldots, A_m$, each representing a distinct architectural perspective such as structural dependencies, semantic relationships and performance metrics. Then, the number of clusters k^* is determined by a weighted evaluation of quality attributes that the resulting microservice architecture must satisfy. This ensures that the decomposition aligns with the prioritized architectural goals. Formally, we aim to solve:

$$\arg\max_{S} \sum_{i=1}^{k^*} \left(Q(S_i) - \lambda \sum_{j \neq i} C(S_i, S_j) \right) \tag{1}$$

where $Q(S_i)$ represents the quality metrics of microservice S_i, $C(S_i, S_j)$ denotes the coupling between services S_i and S_j, λ is a tunable weight and k^* is determined based on the weighted prioritization of quality attributes.

3.2. Multi-View Spectral Clustering Approach

To capture different architectural properties, we define multiple embedding functions, each corresponding to a distinct view:

- **Static analysis** (G_c): Models the static coupling between architectural components by analyzing function calls, inter-module dependencies and invocation patterns. This perspective enables the identification of cohesive modules and tightly coupled elements that influence decomposition decisions.
- **Component semantics** (G_d): Captures domain-driven relationships between architectural elements by employing Natural Language Processing (NLP) techniques (e.g., identifier tokenization, documentation analysis). This view facilitates the grouping of components based on conceptual similarity and domain alignment within the architecture.
- **Dynamic analysis** (G_p): Encodes the execution characteristics as the dynamic behavior of the monolith. This view provides insights user-driven architectural decisions.

Each graph G_i represents a different perspective of the monolith, but it is not necessarily limited to these perspectives. The framework allows the aggregation of other types of inputs, provided they can be vectorized, such as historical development commits, pull request information and team ownership structures. This flexibility ensures a comprehensive and adaptable partitioning strategy.

We define a weighted multi-view affinity matrix A as:

$$A = \sum_{i=1}^{m} \omega_i A_i \tag{2}$$

where A_i represents the affinity matrix corresponding to the i^{th} view and ω_i is a tunable weight that adjusts the importance of each perspective [Chao et al. 2021].

This matrix captures inter-component relations specific to the perspective of view V_i. To ensure consistency for spectral processing, each affinity matrix A_i is symmetrized to guarantee $A_i = A_i^\top$ and subsequently used to compute the corresponding normalized Laplacian matrix $\mathcal{L}_i = I - D_i^{-1/2} A_i D_i^{-1/2}$, where D_i is the degree matrix of A_i.

Spectral clustering is then applied on A, using its eigenvectors to project the components into a lower-dimensional space where clustering is performed. The partitioning S is obtained by applying K-Means to the eigenvector representation of A.

Algorithm 1 Multi-View Spectral Clustering for Microservices Decomposition

Require: Graph views $G_c, G_d, G_p, \ldots, G_m$, weight parameters $\omega_c, \omega_d, \omega_p, \ldots, \omega_m$
Ensure: Partitioning $S = S_1, S_2, ..., S_k$

1: Compute affinity matrices $A_c, A_d, A_p, \ldots, A_m$ from $G_c, G_d, G_p, \ldots, G_m$ respectively.
2: Construct the weighted multi-view affinity matrix $A = \sum_{i=1}^{m} \omega_i A_i$.
3: Determine the optimal number of clusters $k*$ by maximizing $Q(S)$.
4: Compute the eigenvectors of A and project the components into the eigen-space.
5: Apply K-Means clustering on the eigenvector representation with $k*$ clusters.
6: Return optimal partitioning S.

3.3. Optimal Cluster Selection

To determine the optimal number of clusters k, we adopt a quality-driven selection method based on a weighted combination of key architectural quality attributes. For a given clustering solution S, the overall quality score is defined as:

$$Q(S) = \omega_{\text{coh}} \cdot \text{Coh}(S) - \omega_{\text{coup}} \cdot \text{Coup}(S) - \omega_{\text{comp}} \cdot \text{Comp}(S), \tag{3}$$

where:

- $\text{Coh}(S)$ denotes the average intra-cluster cohesion. Measures the internal semantic similarity among components within the same cluster. Higher values indicate better modular grouping.
- $\text{Coup}(S)$ represents the inter-cluster coupling. Captures the semantic or structural similarity between components across different clusters. Lower values are preferred.
- $\text{Comp}(S)$ quantifies the overall complexity of the decomposition (e.g., measured in lines of code). Represents the aggregated implementation cost (e.g., sum of lines of code per cluster). Lower values imply simpler, more manageable services.

The weights $\omega_{\text{coh}}, \omega_{\text{coup}}, \omega_{\text{comp}} \in [0, 1]$ satisfy the constraint $\omega_{\text{coh}} + \omega_{\text{coup}} + \omega_{\text{comp}} = 1$, enabling dynamic prioritization of architectural quality attributes based on design goals. The proposed framework is flexible and can be extended to support alternative evaluation strategies, such as Pareto-based optimization or the inclusion of additional quality attributes.

The optimal number of clusters k^* is obtained by maximizing the quality function $Q(S)$ across a candidate range $[k_{\min}, k_{\max}]$:

Algorithm 2 Optimal Cluster Selection via Quality Maximization

Require: Candidate cluster range $[k_{min}, k_{max}]$, quality metric function $Q(S_k)$
Ensure: Optimal number of clusters k^*

```
Initialize best_k ← k_min, best_score ← −∞
for k ← k_min to k_max do
    Compute partitioning S_k using Spectral Clustering with k clusters
    Compute quality score Q(S_k)
    if Q(S_k) > best_score then
        best_score ← Q(S_k)
        best_k ← k
    end if
end for
Return best_k
```

$$k^* = \arg \max_{k \in [k_{\min}, k_{\max}]} Q(S_k), \tag{4}$$

where S_k denotes the clustering solution obtained with k clusters. The selection process iteratively evaluates each candidate k by computing its corresponding quality score using Equation 3.

4. Evaluation and Results

To assess the effectiveness of the proposed multi-view spectral clustering approach for monolith-to-microservices decomposition, we conducted a preliminary evaluation using a representative set of software components extracted from a real-world e-commerce codebase, consisting of 18 parent classes, each containing approximately 4 to 5 methods.

To simulate the decision-making process of a software architect, we iteratively evaluate the clustering results over a grid of weight combinations $(w_{\text{coh}}, w_{\text{coup}}, w_{\text{comp}})$, reflecting different trade-offs among quality attributes. For each combination, we compute evaluation metrics and select clustering configurations accordingly, as illustrated in Figure 3. The optimal number of clusters is dynamically determined by maximizing a weighted quality score that combines cohesion, coupling and complexity.

The results indicate that, for this repository and the input views considered, increasing the number of services does not lead to a substantial reduction in coupling, although it may yield marginal improvements in cohesion. Notably, there appears to be a trade-off point where high cohesion is attained with relatively low complexity, suggesting that moderate values of k may provide a balanced and effective decomposition. In contrast, excessive fragmentation—reflected in high complexity—can adversely affect modularity and maintainability. Moreover, the observed coupling remains largely unaffected by the granularity of the decomposition. This behavior may result from persistent shared dependencies or tightly coupled responsibilities that span across service boundaries. Additionally, the optimal number of clusters $k* = 18$, obtained under uniform weighting of the input views and quality attributes, appears to be strongly influenced by the structure of the original monolith rather than by the intended post-decomposition objectives. Conse-

quently, more sophisticated decomposition strategies—such as those guided by domain-driven design principles—may be necessary to achieve meaningful decoupling.

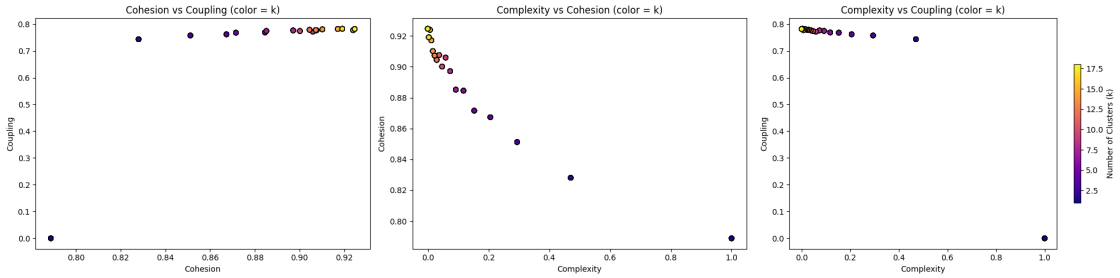

Figure 3. The three scatter plots illustrate the interplay between architectural quality attributes — cohesion, coupling and complexity — under varying numbers of clusters k (potential microservices). The color of each point indicates the value of k, which serves as a proxy for microservice granularity.

5. Discussion & Conclusion

The experimental results demonstrate the potential of multi-view spectral clustering to support monolith decomposition by integrating structural, semantic and runtime-based views. This multi-perspective approach offers software architects a more informed basis for identifying service boundaries.

However, several limitations remain. The effectiveness of the clustering depends heavily on the appropriate weighting of affinity matrices, which requires domain expertise. Additionally, the incorporation of dynamic behavioral data—such as execution traces and request flow patterns—could further improve boundary accuracy. Scalability also requires further evaluation, particularly for large-scale enterprise systems with thousands of components.

Future work should explore adaptive weight tuning, dynamic component reconfiguration during migration and guidance based on technical goals such as scalability—highlighted as critical in recent systematic studies [Martínez Saucedo et al. 2025]. It is also essential to investigate various granularity levels and benchmark the approach against state-of-the-art methods and datasets.

These improvements will contribute to a more automated, scalable and parametrizable methodology for transitioning monolithic systems into microservices architectures.

References

Andrade, B., Santos, S., and Silva, A. R. (2022). From monolith to microservices: Static and dynamic analysis comparison. arXiv:2204.11844.

Andrade, B., Santos, S., and Silva, A. R. (2023). A comparison of static and dynamic analysis to identify microservices in monolith systems. In Tekinerdogan, B., Trubiani, C., Tibermacine, C., Scandurra, P., and Cuesta, C. E., editors, *Software Architecture*, pages 354–361, Cham. Springer Nature Switzerland.

Asseldonk, L. v. (2021). From a monolith to microservices: the effect of multi-view clustering. Master's thesis, Utrecht University.

Chao, G., Sun, S., and Bi, J. (2021). A survey on multiview clustering. *IEEE Transactions on Artificial Intelligence*, 2(2):146–168.

Fritzsch, J., Bogner, J., Wagner, S., and Zimmermann, A. (2019). Microservices migration patterns: A survey. In *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*.

Gysel, M., Kölbener, L., Giersche, W., and Zimmermann, O. (2016). Service cutter: A systematic approach to service decomposition. *IEEE Software*.

Lago, P. and van Vliet, H. (2005). Explicit assumptions enrich architectural models. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*, pages 206–214.

Martínez Saucedo, A., Rodríguez, G., Gomes Rocha, F., and dos Santos, R. P. (2025). Migration of monolithic systems to microservices: A systematic mapping study. *Information and Software Technology*, 177:107590.

Mazlami, G., Cito, J., and Leitner, P. (2017). Extraction of microservices from monolithic software architectures. In *Proceedings of the 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*.

Papazoglou, M. P. and van den Heuvel, W.-J. (2007). Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415.

Richardson, C. (2018). *Microservices Patterns: With Examples in Java*. Manning Publications.

Taibi, D. and Systä, K. (2019). From monolithic systems to microservices: A decomposition framework based on process mining. In *International Conference on Cloud Computing and Services Science*.

Zhou, L., Du, G., Lü, K., Wang, L., and Du, J. (2024). A survey and an empirical evaluation of multi-view clustering approaches. *ACM Comput. Surv.*, 56(7).