Journal of Software Engineering Research and Development, 2026, 14:22, doi: 10.5753/jserd.2026.7575

Granulify: Continuous Microservice Granularity Management with Saturation Signals from Longitudinal Industrial Evidence

Yan Justino [**CESAR SCHOOL** | ***contact@yanjustino.com***]
Carlos Eduardo da Silva [**Sheffield Hallam University** | ***C.DaSilva@shu.ac.uk***]
Rafael Batista Duarte [**CESAR SCHOOL** | ***rbd@cesar.school***]

Abstract

Problem: Microservice granularity is typically treated as a one-time design decision, yet service boundaries must evolve with changing technical and business demands. The absence of systematic mechanisms for continuously reassessing granularity often leads to excessive fragmentation, increased maintenance effort, and elevated operational costs. **Method:** This paper presents *Granulify*, a novel evidence-based approach for continuous granularity management that addresses critical gaps identified in a systematic mapping of 24 primary studies. The approach integrates qualitative diagnosis and quantitative assessment to support iterative boundary adjustments. The approach introduces the Granularity Classification Spectrum (GCS) for positioning services along a granularity continuum, and the Granularity Saturation Method (GSM), which combines polarity-aligned metrics into two aggregate indices—Net Benefit (NB) and Total Variation (TV)—and derives a bounded Saturation Score (SS) from their ratio for regime classification. Thresholds are calibrated through a lagged rolling window, ensuring that each period is classified only using prior historical observations. We followed an industry-oriented Design Science Research approach, operationalised through 13-month longitudinal action research in a large-scale financial services platform, where the first author served as lead architect actively participating in granularity decisions while systematically collecting evidence. **Results:** The rolling analysis treated the initial months as calibration warm-up and identified the first actionable over-decomposition signal in February 2024, when negative Net Benefit values indicated architectural degradation. These quantitative signals were interpreted through triangulation with qualitative observations from sprint retrospectives and architecture review meetings. During the prospective intervention phase, Granulify supported the team in identifying efficient decomposition periods, detecting empirical saturation before further fragmentation, and performing a strategic consolidation that was associated with the highest Saturation Score values of the observation period. **Conclusion:** Granulify provides actionable, evidence-based support for architectural decisions that traditionally relied on subjective judgement, supporting teams in detecting saturation signals and adjusting service boundaries before incurring unnecessary complexity and costs.

Keywords: *Microservices, Service Granularity, Continuous Architecture Management, Software Metrics, Longitudinal Action Research*

1 Introduction

Precisely establishing the size and scope of a microservice remains one of the main software engineering challenges faced by development teams. Prior studies highlight the complexity involved in granularity decisions and their direct impact on the modularity, performance, maintenance, and evolution of microservice-based systems (Fritzsch et al., 2019; Bogner et al., 2021; Becker and Lucrédio, 2020; Waseem et al., 2021; Colanzi et al., 2021; Zhou et al., 2023; Salii et al., 2023; Hassan et al., 2020; Nunes et al., 2019; Taibi and Lenarduzzi, 2018; Vural and Koyuncu, 2021; Tusjunt and Vatanawood, 2018).

In this regard, granularity decisions are fundamental to software modeling: inadequate decomposition can lead to an excessive number of components and automation mechanisms, which tends to intensify operational complexity and cost, increase the coordination effort required for testing and integration, and raise the likelihood of communication failures and inconsistencies, ultimately increasing maintenance effort over time (Behrad et al., 2021; Ait Said et al., 2024).

Although the literature provides substantial evidence on challenges and strategies for migrating legacy systems to microservices (Fritzsch et al., 2019; Taibi et al., 2017; Colanzi et al., 2021), an important gap remains: granularity should be treated as a dynamic property that evolves with the systems, teams, and business demands; however, there is limited operational guidance on how to continuously reassess and adjust service boundaries after the initial decomposition (Hassan et al., 2020; Vera-Rivera et al., 2021; Bogner et al., 2021; Driessen et al., 2024). In practice, teams still face recurring concerns when managing microservice granularity, including the following:

- **Lack of objectivity and limited tooling support:** service boundary decisions frequently rely on expert judgment, which is not always consistently available or systematically justified (Driessen et al., 2024; Vera-Rivera et al., 2021).
- **Need for dynamic, evidence-driven assessment:** granularity assessment should capture both the expected value and the incurred cost of (re)decomposition decisions, rather than treating granularity as a one-time design choice (Hassan et al., 2021).
- **Lack of actionable guidance for decomposition and recomposition:** many approaches describe principles, but few provide repeatable procedures and instruments that practitioners can apply in real-world contexts (Vera-

Rivera et al., 2021; Zhang et al., 2019).

- **Limited longitudinal industrial evidence:** empirical accounts of granularity adjustment and its consequences over time remain comparatively scarce (Bogner et al., 2021; Hassan et al., 2020).

These concerns arise, in large part, because granularity is frequently treated as a one-time decision during the initial decomposition, rather than as an ongoing management activity. To address this gap, this study presents **Granulify**, an evidence-based approach to *continuous granularity management* in distributed systems, providing guidance on adjustments throughout the application lifecycle through progressive modularization and architectural adaptation.

In this paper, we use the term *continuous granularity management* to denote an iterative process of monitoring, evaluating, and adjusting service boundaries throughout the system lifecycle. This notion differs from *initial decomposition*, which focuses on partitioning the system during the early stages (e.g., during migration, initial design, or modernization) and frequently assumes that the chosen granularity will remain adequate. *Continuous granularity management*, in contrast, emphasizes periodic reassessment based on accumulated evidence of system evolution, aiming to detect *granularity saturation*.

We define *granularity saturation* as the point at which continuous adjustments to service boundaries no longer yield measurable improvements in the system's quality attributes. Recognizing saturation helps teams avoid over-decomposition and the associated operational overhead, focusing instead on architectural stabilization and addressing other concerns.[1]

This approach was developed and evaluated through a 13-month longitudinal industrial study involving the migration and evolution of a large investment platform in the Brazilian financial sector. In this context, Granulify provided systematic guidance for granularity decisions, supporting the team in balancing the benefits of decomposition against operational costs and in detecting when the system had reached an adequate level of granularity.

The main contribution of this study is the definition of an approach for continuous granularity management, including a repeatable granularity review protocol and supporting instruments for mapping and assessment. Specifically, this paper presents the following:

1. The **Granularity Classification Spectrum (GCS)**, a conceptual model that categorizes granularity levels in microservices, facilitating understanding and communication about different decomposition strategies.
2. The **Granularity Saturation Method (GSM)**, which supports granularity saturation assessment by combining a set of polarity-aligned metrics into two complementary aggregate indicators: **Net Benefit** (NB), capturing the directional effect (improvement vs. degradation), and **Total Variation** (TV), capturing the magnitude of structural change (turbulence) regardless of sign.

[1]This differs from its use in Site Reliability Engineering in Google's Golden Signals, where saturation denotes how busy a constrained execution resource is, such as CPU, memory, I/O, or queue capacity.

Furthermore, it presents initial empirical evidence from an industrial study that illustrates the practical application and benefits of Granulify, providing insights into how teams can navigate the complexities of microservice granularity over time. This paper also extends our prior work presented at the 39th Brazilian Symposium on Software Engineering (Justino et al., 2025). The main additions include (i) a comprehensive related work section based on a systematic mapping study of 24 primary studies; (ii) a detailed description of the Granularity Classification Spectrum; (iii) the complete Granularity Saturation Method, with formal definitions and thresholds; (iv) extended empirical evidence from the industrial study; and (v) an in-depth discussion of threats to validity and future work.

The remainder of this paper is organized as follows: Section 2 discusses related work on microservice granularity; Section 3 details the Granulify approach; Section 4 presents the research method and industrial study design; Section 5 presents the results and discussion; and Section 6 concludes the paper with contributions, limitations, and future work.

2 Related Work

This section presents the theoretical foundation underlying this work. It is derived from a systematic mapping study (SMS) of 25 primary studies on microservice granularity published between 2017 and 2024. The analysis of this mapping identifies critical gaps in continuous granularity management and positions the contributions of this research.

2.1 Conceptual Foundations

In a classical conception, granularity relates to the level of detail or abstraction of a software component, reflecting its size, functional scope, and degree of coupling among system units Parnas (1972); Shaw and Garlan (1996). When this notion is brought to microservice-based architectures, granularity also becomes a boundary-definition problem Vera-Rivera et al. (2021).

Delimiting such boundaries remains one of the most challenging decisions in the design and evolution of these systems Hassan et al. (2020); Salii et al. (2023); Vera-Rivera et al. (2021). This is because granularity decisions depend on multiple factors: domain boundaries (Domain-Driven Design) Vural and Koyuncu (2021); Ait Said et al. (2024), maintainability and evolvability Becker and Lucrédio (2020); Taibi and Lenarduzzi (2018), operational infrastructure cost Hassan et al. (2022); Vera-Rivera et al. (2021), and service independence and flexibility Hassan et al. (2020).

In other words, these challenges bring with them a set of trade-offs: finer decomposition may improve modularity and scalability, but may also increase architectural complexity, communication overhead, and operational cost Hassan et al. (2020); Salii et al. (2023). These adverse effects are interpreted in this paper as indicators of a *saturation point*, at which further decomposition or unnecessary aggregation no longer produces net benefits due to coordination and complexity costs.

In this sense, selecting granularity in practice involves significant trade-offs (e.g., scalability vs. cost; modularity vs. coordination; modularity vs. maintainability) on the part of a team Hassan et al. (2020); Driessen et al. (2024); Josélyne et al. (2018); Colanzi et al. (2021).

2.2 Systematic Mapping Protocol

To ensure transparency and rigor, we conducted a systematic mapping study following established guidelines Kitchenham et al. (2007). The search was executed between June and August 2023 in the ACM Digital Library, IEEE Xplore, and ScienceDirect, using the query *("micro service" OR microservice) AND (granularity OR size OR decomposition)*. The automated search returned 1,749 records. After duplicate removal, 1,366 unique studies remained. Sequential screening (title, abstract, and full-text assessment) resulted in **23 primary studies** for synthesis.

To update the evidence base beyond the original search window, we conducted a complementary *snowballing* procedure from the initially mapped corpus, focusing on publications from 2024 and 2025. The update combined forward *snowballing*, to identify recent studies citing the mapped corpus, and backward *snowballing*, to inspect the references of newly identified candidates.

Across the entire screening (52 candidates assessed), 25 studies were accepted (acceptance rate: 48.1%), incorporating recent work on quantitative granularity assessment, industrial microservice adoption, decomposition modeling, and temporal indicators of architectural degradation Driessen et al. (2024); Ait Said et al. (2024); Bakhtin et al. (2025); Bakhtin (2025).

The 2025 studies were used as adjacent evidence for positioning, rather than as primary SMS inputs, because they address architectural degradation detection through temporal network analysis, rather than granularity decision-making. Overall, the *snowballing* update reinforced the relevance of microservice granularity as an active research topic, but did not alter the main gap identified in the mapping: no study in the final corpus combines continuous granularity management, longitudinal industrial evidence, an operational methodological instrument, and an explicit temporal saturation signal.

After selection, we conducted a systematic classification of each of the 25 primary studies across bibliographic, methodological, contextual, and technical dimensions. The results of this classification are synthesized in the next subsection, followed by a gap analysis that motivates the contributions of this paper.

2.3 Evidence Synthesis

This subsection synthesizes evidence from the 25 primary studies by examining (*i*) the Study Profile and Methodological Landscape, (*ii*) Contribution and Artifact Patterns, (*iii*) Granularity Actions Addressed (e.g., decomposition, migration, merging, and splitting), (*iv*) Assessment Criteria and Metric Families used to evaluate granularity, (*v*) Lifecycle Coverage and Saturation Support, and (*vi*) the coverage of architectural forms across the Granularity Spectrum (G0–G6).

2.3.1 Study Profile and Methodological Landscape

The 25 studies are distributed across journals (48.0%), conferences (48.0%), and other publication types, such as theses (4.0%). [2] The main publication venues include IEEE ICSA, IEEE Software, Journal of Systems & Software, and ACM conferences. Research methods are dominated by case studies (52.0%) and surveys (24.0%), followed by studies without empirical evaluation (16.0%) and mixed methods (8.0%). An industrial context is present in 64.0% of the studies; laboratory/academic environments account for 16.0%, open-source projects for 12.0%, and the context is not reported in 8.0%.

2.3.2 Contribution and Artifact Patterns†

The primary contribution types include methods or processes (28.0%), catalogs or taxonomies (16.0%), metrics or measures (8.0%), tools (8.0%), guidelines or checklists (4.0%), and frameworks (4.0%). Notably, 56.0% of the studies provide some form of reusable operational instrument. In terms of availability, 52.0% of the studies describe their instruments only in the paper, 32.0% provide public links or repositories, and 4.0% report instruments that are unavailable.

2.3.3 Granularity Actions Addressed†

Decomposition is the dominant action addressed (68.0%), followed by migration (64.0%) and merging (36.0%). Splitting (32.0%), boundary refactoring (20.0%), and smell/antipattern detection (20.0%) also receive notable attention, indicating that the corpus covers a broader range of boundary operations than is commonly recognized.

2.3.4 Assessment Criteria and Metric Families†

The literature recognizes the multidimensional nature of granularity decisions: 44.0% of the studies employ explicit multi-criteria analysis. The most frequently used metric families are coupling (76.0%), maintainability (68.0%), cohesion (44.0%), scalability (40.0%), domain alignment (40.0%), size (40.0%), team/organizational factors (28.0%), performance (28.0%), reliability (16.0%), complexity (16.0%), and cost (16.0%). Criterion combination approaches are distributed among heuristics (28.0%), rule-based approaches (24.0%), structured qualitative analysis (16.0%), and weighted scoring (4.0%); the combination approach is not reported in 28.0% of the studies.

2.3.5 Lifecycle Coverage and Saturation Support†

Regarding lifecycle stages, multiple/end-to-end stages (40.0%) and design (36.0%) are the most represented, followed by evolution (28.0%) and operation (4.0%).

[2] Methodological note: in systematic mapping studies, several dimensions are multi-label (a single study may fall into more than one category), such as metric families, granularity actions, lifecycle stages, and sometimes context and contribution type; this also applies to the granularity spectrum classification (G0–G6) introduced in this section. In such cases, the reported percentage represents coverage, so percentages may exceed 100% when summed. Multi-label dimensions are marked with † in this section.

2.3.6 Coverage of Granularity Spectrum Levels (G0–G6)†

To characterize how the evidence base frames architectural granularity, we classified each primary study into the categories of the *Granularity Classification Spectrum* (G0–G6) Justino et al. (2025). Table 1 summarizes the resulting distribution. [3] In this paper, we detail each spectrum level below, extending definitions from prior work Justino et al. (2025). In Section 3, we operationalize these levels within the Granulify approach.

Table 1. Coverage of the 25 primary studies across the Granularity Classification Spectrum (G0–G6).

Level	Category	N	Coverage
G0	Monolith	07	28.00%
G1	Modular Monolith	01	4.00%
G2	Service-Based	04	16.00%
G3	Extended Microservice	25	100.00%
G4	Intermediate Microservice	14	56.00%
G5	Focused Microservice	05	20.00%
G6	FaaS / Nanoservice	01	4.00%

Overall, the evidence anchors granularity discussions in microservice contexts (G3) and, in most cases (56.00%), also discusses intermediate microservice boundaries (G4), i.e., services with sub-capability or subdomain scope. In contrast, focused microservices (G5) appear in a minority of studies (20.00%), frequently in the context of risks and corrective actions (e.g., avoiding over-segmentation and merging excessively fine-grained services). References to monoliths (G0) appear in 28.00% of the studies, typically as the migration origin.

Service-based architectures (G2, 16.00%) appear both as a transitional architectural form in migration studies Huang et al. (2020); Tusjunt and Vatanawood (2018); Bogner et al. (2019) and, in one case, as the *Mega Service* antipattern Tighilt et al. (2023), representing a coarse-grained boundary violation that the evidence base explicitly flags as problematic.

A single primary study addresses each of the modular monolith (G1) and serverless/nanoservice (G6) perspectives, showing that these boundary forms remain underexplored in the evidence base.[4]

2.4 Gap Analysis and Positioning

Based on the preceding synthesis, we positioned the 25 primary papers against five binary criteria that capture the dimensions most relevant to this study: **Continuous Granularity Management** (CGM), **Longitudinal Data Analysis** (LDA), **Industrial Application Context** (IAC), **Operational/Methodological Instrument** (OMI), and **Temporal Saturation Signal** (TSS).

The coding scheme combines empirical, artifactual, and gap-oriented dimensions derived from the SMS synthesis. LDA and IAC distinguish longitudinal evidence from cross-sectional analysis and industrial validation. OMI captures whether a study provides an operational method, tool, checklist, protocol, or reusable instrument. CGM captures whether granularity is treated as an ongoing lifecycle concern, rather than a one-time decomposition or refactoring decision.

Finally, TSS was introduced to identify studies that define an explicit temporal signal for detecting saturation or stabilization in granularity evolution. Table 2 summarizes this positioning. The criteria are used as an analytical coding scheme for gap identification, rather than as a quality ranking of the primary studies.

The evidence base shows a clear asymmetry. Although 44.0% of the studies employ explicit multi-criteria analysis—indicating that the field increasingly recognizes the multi-dimensional nature of granularity decisions—this analytical breadth is generally applied to point-in-time assessments, rather than to continuous management or temporal saturation detection. Examining these dimensions jointly, the resulting gaps are:

- **Continuous Granularity Management (4.00%)**: Only one study Hassan et al. (2022) proposes a mechanism for continuous granularity management throughout the system lifecycle.
- **Longitudinal Data Analysis (8.00%)**: Only 2 studies analyze temporal data at multiple points; 92.0% perform cross-sectional analysis (single snapshot).
- **Operational Guidance for Reassessment (56.00%)**: Although the studies provide an operational or methodological instrument (OMI), these instruments generally do not define explicit criteria for when granularity should be reassessed. The predominant approach treats granularity as a *point-in-time decision* (at design time or eventual refactoring), rather than as a *continuous process* that evolves with the system.
- **Temporal Saturation Signal (0%)**: No study defines an indicator for determining when granularity has reached a stabilization or saturation point—the most critical gap identified.

Beyond these identified gaps, recent work on architectural degradation provides an adjacent but distinct line of evidence. The study by Bakhtin et al. Bakhtin et al. (2025) analyzes microservice degradation through temporal network structures and centrality-change indicators, showing that architectural decay can be detected from evolving dependency graphs.

This work is relevant to our positioning because it reinforces the need for temporal reasoning in microservice evolution. However, its focus is on architectural degradation detection, not granularity decision-making: it does not define saturation as a balance between the benefits and costs of decomposition, nor does it provide an operational signal for deciding when to split, merge, or stop decomposing services.

The studies most closely focused on granularity are those by Hassan et al. Hassan et al. (2022) and Driessen et

[3] The categories are not mutually exclusive: a single study may mention multiple architectural forms (e.g., monolith, service-based, and microservices) when discussing adoption, migration, and evolution.

[4] G1 (modular monolith) appears in one primary study Ait Said et al. (2024) and G6 (FaaS/nanoservices) in one study Tighilt et al. (2023), where it is treated as an antipattern (Nano Service). Both levels are also discussed in the gray literature (e.g., technical reports and industry blogs) and in secondary studies (systematic reviews) not covered by the SMS protocol.

Table 2. Systematic mapping of 25 primary studies against dimensions relevant to continuous granularity management (CGM: Continuous Granularity Management; LDA: Longitudinal Data Analysis; IAC: Industrial Application Context; OMI: Operational/Methodological Instrument; TSS: Temporal Saturation Signal).

ID	Study	Continuous Granularity Management	Longitudinal Data Analysis	Industrial Application Context	Operational/ Methodological Instrument	Temporal Saturation Signal
p01	Taibi et al. (2017)	-	-	✓	✓	-
p02	Josélyne et al. (2018)	-	-	-	✓	-
p03	Taibi and Lenarduzzi (2018)	-	-	✓	-	-
p04	Tusjunt and Vatanawood (2018)	-	-	-	✓	-
p05	Bogner et al. (2019)	-	-	✓	-	-
p06	Cruz et al. (2019)	-	-	✓	✓	-
p07	Fritzsch et al. (2019)	-	-	✓	-	-
p08	Nunes et al. (2019)	-	-	-	✓	-
p09	Zhang et al. (2019)	-	-	✓	-	-
p10	Becker and Lucrédio (2020)	-	-	✓	-	-
p11	Huang et al. (2020)	-	-	✓	-	-
p12	Behrad et al. (2021)	-	-	-	✓	-
p13	Bogner et al. (2021)	-	-	✓	-	-
p14	Colanzi et al. (2021)	-	-	✓	✓	-
p15	Gravanis et al. (2022)	-	-	-	✓	-
p16	Mendonca et al. (2021)	-	-	✓	-	-
p17	Vural and Koyuncu (2021)	-	-	-	✓	-
p18	Waseem et al. (2021)	-	-	✓	-	-
p19	Hassan et al. (2022)	✓	-	-	✓	-
p20	Zhong et al. (2022)	-	✓	✓	✓	-
p21	Hasan et al. (2023)	-	-	-	✓	-
p22	Tighilt et al. (2023)	-	-	-	✓	-
p23	Zhou et al. (2023)	-	-	✓	-	-
p24	Driessen et al. (2024)	-	✓	✓	✓	-
p25	Ait Said et al. (2024)	-	-	✓	-	-
	This work	✓	✓	✓	✓	✓
	Coverage	4.00%	8.00%	64.00%	56.00%	0.00%

al. Driessen et al. (2024), but they differ from Granulify in four important respects. First, in terms of artifact, Hassan et al. provide a scalability analysis approach for granularity adaptation decisions, while Driessen et al. provide a quantitative maintainability assessment method for identifying services as candidates for merging or decomposition; Granulify combines a granularity classification spectrum, a saturation assessment method, and a repeatable management process.

Second, in terms of assessment method, those studies evaluate specific quality perspectives—scalability in Hassan et al. and maintainability through coupling, cohesion, size, and change-related metrics in Driessen et al.—whereas Granulify aggregates polarity-aligned indicators across architectural, development, deployment, code quality, and operational dimensions.

Third, in terms of application context, Driessen et al. validate their method through selected refactoring cases in mixed real-world and open-source systems, while Granulify is evaluated through 13 months of longitudinal action research in a large-scale financial services platform.

Finally, in terms of the generated signal, the prior works produce quality-oriented assessments or refactoring candidates, but not a temporal saturation signal that classifies whether the observed granularity changes represent efficient decomposition, empirical saturation, over-decomposition, or consolidation.

This gap analysis highlights the absence of systematic mechanisms for continuous granularity management and the lack of empirical evidence on granularity saturation signals. Addressing these gaps represents a significant research opportunity that this work directly tackles.

2.5 Gap Statement

The evidence base demonstrates substantial progress in (*i*) defining and measuring granularity through multiple criteria, (*ii*) cataloging symptoms and decision heuristics (e.g., architectural smells and decomposition guidance), and (*iii*) discussing trade-offs and quality attributes. However, three gaps remain particularly salient for practice:

1. The lack of a **continuous and repeatable method** for revisiting granularity decisions throughout the system lifecycle (only 4.00% address this);
2. The absence of a **temporal saturation signal** that supports longitudinal reasoning about when granularity stabilization has been achieved (0% address this);
3. The limited operationalization of **reassessment triggers** in existing instruments, i.e., even when operational guidance is provided, the criteria for deciding when granularity should be reassessed generally remain implicit.

Granulify was designed to directly address these gaps, combining continuous monitoring, longitudinal evidence collec-

tion, operational instruments for granularity assessment, and explicit temporal saturation signals to guide adjustment decisions in industrial environments. The next section presents the Granulify method, detailing how these elements are organized into a practical approach for evaluating service granularity and supporting adjustment decisions.

3 Granulify Method

This section details the Granulify method, an evidence-based approach for continuous granularity management in distributed systems. The method is structured around an iterative cycle of granularity assessment and adjustment, supported by conceptual artifacts and operational instruments that facilitate informed decision-making throughout the system lifecycle.

In the following subsections, we describe the guiding principles of the method, the Granularity Classification Spectrum (GCS) used to map services along a granularity continuum, and the Granularity Saturation Method (GSM) for assessing when granularity has reached a saturation point. These elements form the foundation of the iterative process that teams can follow to monitor and adjust the granularity of their systems in a systematic, evidence-based manner.

3.1 Principles and Process

Granulify has a set of guiding principles and an iterative macroprocess for granularity assessment and adjustment. These elements are designed to support teams in navigating the inherent trade-offs in granularity decisions, providing a structure for continuous monitoring, evidence-based assessment, and informed decision-making. At the principles level, Granulify emphasizes:

i **Granularity as a spectrum of manifestations:** service granularity should be viewed along a spectrum, ranging from coarse-grained (monolith-like) to fine-grained (highly fragmented). The optimal level of granularity varies according to technical constraints, organizational structure, and business needs.

ii **Continuous monitoring:** granularity should be constantly monitored and adapted to balance modularity, performance, and management complexity, taking into account changes in technical and organizational demands.

iii **Evidence-based decision-making:** architectural adjustments should be grounded in both qualitative insights (e.g., domain alignment, team feedback) and quantitative metrics (e.g., coupling, cohesion, change frequency) to ensure that decisions are well-informed and contextually relevant.

At the process level, Granulify is structured as an evidence-driven macroprocess composed of three main phases, connected by explicit decision points and feedback loops. Figure 1 summarizes the iterative cycle and how the steps relate to the three-phase macroprocess described above.

The process begins with **(1) Granularity Spectrum Mapping**, in which the existing microservice architecture is analyzed and positioned along a granularity spectrum using the Granularity Classification Spectrum (GCS). After identifying the services, a set of metrics is collected for each service. This evidence is then analyzed to identify potential granularity imbalances, such as excessively coupled services, low cohesion, or services that frequently require coordinated changes.

This analysis produces a Granularity Mapping Report, which consolidates structural evidence. Based on this report, a first decision point assesses whether there are signs of granularity imbalance. If no evidence is identified, the process ends without architectural changes.

The process then advances to **(2) Measuring Granularity Saturation**, where time-series metrics are analyzed through the Granularity Saturation Method (GSM) to assess the observed saturation level. The outcome of this analysis is formalized as a Granularity Saturation Signal, which feeds a second decision point indicating whether an architectural review is needed.

If an adjustment is deemed necessary, the process proceeds to **(3) Redefining the Microservice Architecture**. In this phase, explicit architectural decisions—such as splitting, merging, or reorganizing microservices—are made and documented through an Architecture Decision Record (ADR). These decisions result in a revised architecture, which can subsequently re-enter the assessment cycle.

If at any decision point it is concluded that no adjustment is needed, the process ends preserving the current architecture. Overall, this macroprocess was designed to be iterative and can be executed either periodically (e.g., during regular architecture review cycles) or in response to events, whenever significant architectural changes or strong symptoms of granularity imbalance motivate reassessment.

In this paper, we focus on the first two phases of the process, which were applied and evaluated in the context of the longitudinal study described in Section 4.2. The next subsections detail the artifacts and methods used in these phases.

3.2 Granularity Spectrum Mapping

The goal is to identify and position existing services along a granularity continuum, collecting their metrics, mapping the system topology, and assessing potential misalignments and areas for improvement. This mapping is performed with the support of the *Granularity Classification Spectrum (GCS)* artifact. The composition of the service map and the associated service metrics forms the *Granularity Mapping Report*, which serves as a communication and analysis artifact.

Below, we describe the GCS and the classification criteria used to position services along the granularity spectrum. We also present a set of structural metrics used to assess services and identify potential granularity imbalances, as well as the indicators that guide analysis and decision-making.

3.2.1 Granularity Classification Spectrum (GCS)

Granulify introduces the GCS, a conceptual model for distinguishing different levels of granularity. Figure 2 presents

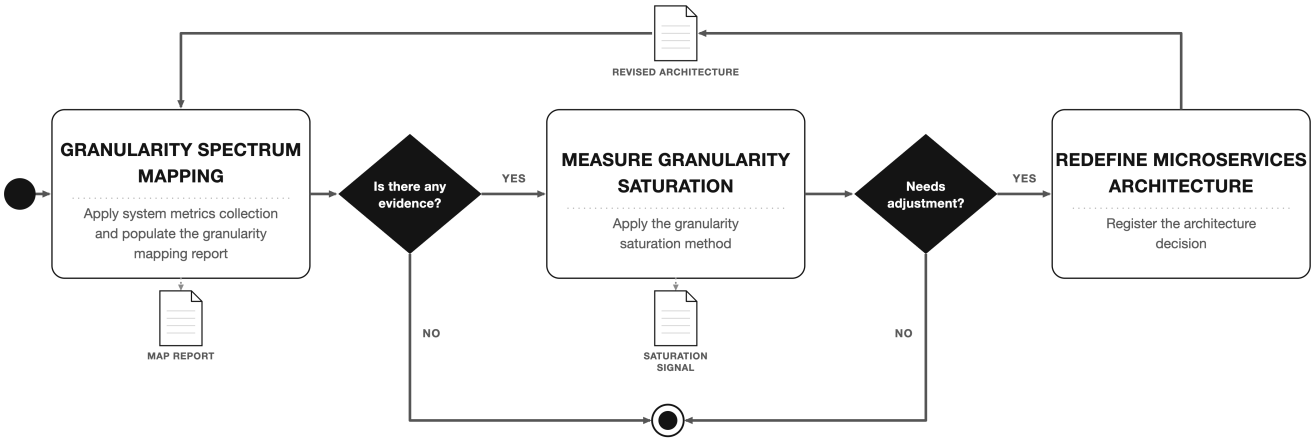

Figure 1. Granulify Macroprocess for evidence-based assessment and refinement of microservice granularity. *Source: the authors' own work.*

a visual representation of the GCS, which organizes architectural styles along a continuum ranging from monoliths to nanoservices, as well as the heuristic criteria used to map deployment artifacts to GCS levels.

At the far left, granularity integrators—Monolithic Application (G0), Modular Monolith (G1), and Service-Based Architecture (G2)—consolidate multiple business capabilities into a single deployable component. As functional decomposition deepens, the system progresses through three microservice subclasses: Extended (G3), representing a single business capability; Intermediate (G4), a cohesive grouping of related functionalities; and Focused (G5), a single functionality.

This progressive refinement of service boundaries culminates in the rightmost level—the nanoservice [5] or *FaaS* (G6)—in which each ephemeral, event-driven unit encapsulates a single business function.

3.2.2 Classification Criteria

Complementary to the visual spectrum, the GCS is operationalized through a set of classification criteria that allow positioning architectural artifacts along the granularity continuum. This classification is based on seven heuristic criteria that consolidate recurring characteristics observed in the industrial literature and commonly used in architectural reviews Newman (2021); Ford (2023); Fowler (2017); Richards (2015); Vernon (2015); Evans (2003). Each of the following criteria captures a distinct architectural dimension:

- **Deployment unit** describes the packaging and deployment boundary: *single* (entire system deployed as one unit); *multiple* (multiple independently deployable components); *autonomous* (each service is an autonomous deployment unit); *function* (individual functions as deployment units).
- **Internal modularity** indicates whether the internal structure is explicitly designed and maintained: *non-explicit* (no enforced boundaries), *explicit* (well-defined modules with stable interfaces).
- **Deployment independence** reflects the degree of autonomy in deployment decisions: *none* (cannot be deployed separately), *partial* (limited autonomy), *full* (complete deployment autonomy).
- **Data ownership** describes how data is managed and accessed: *shared* (common database or data structures), *owned* (each service owns its data), *minimal/none* (stateless or minimal local state).
- **Coupling** characterizes the degree of dependency between components: *strong* (high interdependency); *medium* (moderate dependencies); *low* (minimal dependencies); *very low* (near-zero dependencies).
-
- **Functional scope** indicates the scope of business functionality encapsulated. Values: *entire system* (whole system); *multiple capabilities* (several business areas); *single capability* (one business area); *sub-capability/subdomain* (part of a capability); *single function* (one operation); *atomic event*.
- **State management** indicates how state is handled during execution. Values: *stateful* (maintains persistent state), *predominantly stateful* (predominantly stateful, with some stateless components), *predominantly stateless* (predominantly stateless), *stateless/ephemeral* (no persistent state, exists only during execution).

These criteria provide a structured lens for evaluating architectural artifacts and positioning them along the granularity continuum. The specific combination of values across these seven dimensions is what distinguishes one granularity level from the next. For example, the shift from shared data to owned data, combined with full deployment independence, marks the transition from G2 (Service-Based) to G3 (Extended Microservice).

Before detailing each granularity level, it is important to clarify the terminology used to describe functional scope. We adopt a hierarchical view of functional decomposition aligned with Domain-Driven Design principles Newman (2021): a **business capability** (or simply *capability*) represents a coarse-grained area of business functionality (e.g., *Investment Portfolio Management*); a **sub-capability** (or *subdomain*) represents a more focused functional area within a capability (e.g., *Portfolio Valuation*); a **function** represents

[5] The term nanoservice is frequently cited in the literature as an antipattern Tighilt et al. (2023); Vural and Koyuncu (2021); in our approach, the term is also used to classify a granularity level finer than a microservice.

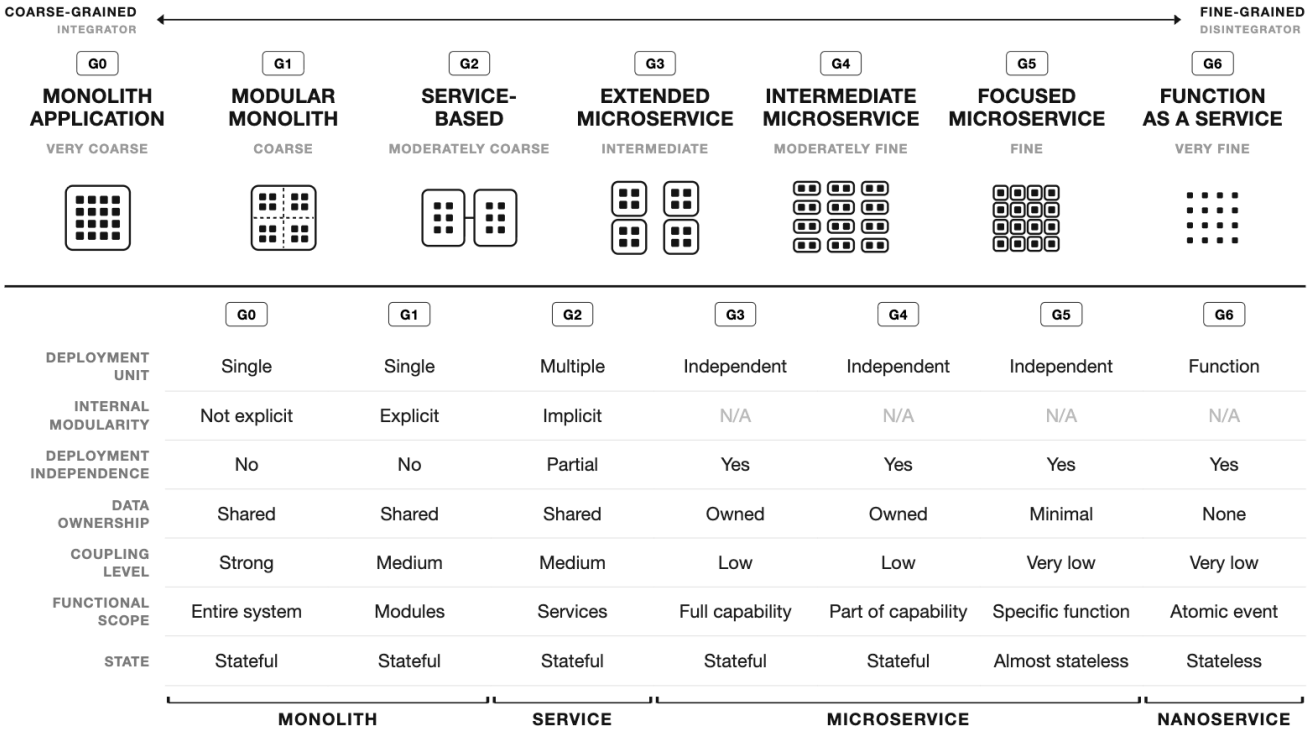

Figure 2. Granularity Classification Spectrum (GCS) Model. Each level represents a distinct granularity pattern, characterized by deployment unit, functional scope, and data ownership. Filled boxes represent individual deployment units (services, functions), while internal subdivisions indicate functional decomposition within a deployment boundary. *Source: the authors' own work.*

a single cohesive operation or responsibility (e.g., *Calculate Net Asset Value*); and an **action** or **event** represents a specific invocation or trigger of a function (e.g., *On Portfolio Update*). This hierarchy guides the classification of services along the spectrum.

3.2.3 Granularity Levels

Below, we detail each level of the Granularity Classification Spectrum, presenting its architectural characteristics, advantages, disadvantages, and associated references. The levels are organized from coarsest (G0) to finest (G6), reflecting a progressive increase in deployment independence, modularity, and functional focus.

The advantages and disadvantages reported for each level are not adopted as a ready-made taxonomy from a single source; they constitute an authorial synthesis derived from the cited literature, the GCS classification criteria, and recurring trade-offs observed in architectural reviews. The references associated with each level indicate the evidence base that informed this synthesis, while the final wording and comparative framing are contributions of this work.

G0 - Monolithic Application (Very Coarse): A monolithic application is structured as a single deployable unit in which internal modularity is not explicitly treated as an architectural decision boundary. With no deployment independence, the system operates with shared data structures, strong coupling between components, and broad business capabilities. The functional scope spans the entire system, and execution is inherently stateful Becker and Lucrédio (2020); Colanzi et al. (2021); Fritzsch et al. (2019); Mendonca et al. (2021).

- **Advantages:** simplicity of deployment and initial testing. Because the application is a single unit, the deployment process is straightforward and less complex. End-to-end testing is also simpler to execute, as there is no need to manage multiple services or network dependencies.
- **Disadvantages:** limited scalability and maintenance difficulties. The application can only be scaled as a whole, even if only a small part of it requires more resources, leading to resource waste. Over time, the codebase can become so intertwined that maintenance and the introduction of new features become slow and risky.

G1 - Modular Monolith (Coarse): A modular monolith remains a single deployable unit but introduces explicit internal modularity through stable interfaces between modules. While still lacking deployment independence, this architecture exhibits medium coupling through well-defined module boundaries, typically operates with shared data, and aligns with multiple business capabilities at the module level. Execution remains stateful Ford (2023); Newman (2021); Ait Said et al. (2024).

- **Advantage:** better organization and maintainability. Internal modularity facilitates code comprehension and division of labor between teams. Explicit boundaries between modules help prevent accidental coupling, making the application easier to maintain and evolve than a traditional monolith.
- **Disadvantage:** lack of deployment autonomy and scalability. Despite the internal organization, the application

remains a single deployment unit. A failure in one module can bring down the entire system, and scalability remains monolithic, with no possibility of scaling modules individually.

G2 - Service-Based (Moderately Coarse): Service-based architectures introduce multiple deployable units with partial deployment independence, although autonomy remains limited. These architectures typically exhibit shared data across services and maintain strong-to-medium coupling. Each service unit commonly spans multiple business capabilities or coarse-grained domains, with execution remaining predominantly stateful Bogner et al. (2019); Huang et al. (2020); Tusjunt and Vatanawood (2018).

- **Advantage:** balance between cohesion and distribution. It allows a smoother transition from a monolith to a distributed architecture. Grouping business capabilities into larger services simplifies communication and data management compared to finer-grained microservices, reducing initial operational complexity.
- **Disadvantage:** high coupling and low separation of concerns. Since services represent multiple capabilities, coupling between them tends to be strong. A change in one business capability may require modifications across multiple services, and the separation of responsibilities is still limited, hindering independent evolution.

G3 - Extended Microservice (Intermediate): Extended microservices are independently deployable units that own their data and align with a single business capability. This level introduces true service autonomy, with low coupling between services, while maintaining stateful execution patterns Ait Said et al. (2024); Becker and Lucrédio (2020); Bogner et al. (2021); Driessen et al. (2024); Zhou et al. (2023).

- **Advantage:** development and deployment autonomy per business capability. Teams can develop, deploy, and scale their services independently, focusing on a complete business capability. This increases agility and allows each team to choose the technology best suited to their needs.
- **Disadvantage:** complexity in integration and data management. Autonomy requires well-defined integration points. Managing data consistency across services and handling distributed transactions become significant challenges that do not exist in monolithic architectures.

G4 - Intermediate Microservice (Moderately Fine): Intermediate microservices are independently deployable units that own their data and align with a sub-capability or subdomain within a broader business capability. These services maintain low coupling with other components, preserving stateful execution Bogner et al. (2021); Driessen et al. (2024); Homay et al. (2019); Waseem et al. (2021); Zhong et al. (2022).

- **Advantage:** greater functional focus and cohesion. By encapsulating functional subsets, each service becomes more specialized and cohesive. This facilitates development and maintenance, as the scope of each service is smaller and better defined than at the full business capability level.
- **Disadvantage:** increased internal dependencies and network communication. Higher granularity leads to a greater number of services that need to communicate to carry out a business operation. This increases latency, network complexity, and the need for robust communication and service discovery mechanisms.

G5 - Focused Microservice (Fine): Focused microservices are independently deployable as persistent services (typically containers or pods) that own minimal state or data. Each service aligns with a single function and exhibits very low coupling with other components. The deployment model is continuous execution (always-on), and these services frequently require orchestration to fulfill end-to-end business flows Behrad et al. (2021); Becker and Lucrédio (2020); Driessen et al. (2024); Taibi and Lenarduzzi (2018).

- **Advantage:** maximum reusability and granular scalability. Small services focused on a single functionality can be easily reused in different business contexts. In addition, they allow extremely precise scalability, allocating resources only to the functionalities that actually need them.
- **Disadvantage:** high operational complexity (overhead). Managing a large number of small services dramatically increases the complexity of deployment, monitoring, logging, and security. The "architectural tax" for keeping the system running becomes very high.

G6 - FaaS (Very Fine): Function-as-a-Service (FaaS) represents function-shaped deployment units deployed on serverless platforms and executed on demand. These functions are deployment-independent by design, maintain no data ownership, and exhibit very low coupling. Each function implements a single event-triggered or invocation-driven operation, operating under an ephemeral, stateless deployment model that is instantiated on demand and terminated after execution Ford (2023); Kolny (2023); Tighilt et al. (2023).

- **Advantage:** cost efficiency and automatic scalability. The billing model is pay-per-use, eliminating costs associated with idle resources. The serverless platform manages all infrastructure and automatically scales functions from zero to thousands of executions as needed.
- **Disadvantage:** strong vendor dependency (vendor lock-in) and execution limitations. The application becomes strongly dependent on the platform and specific APIs of the cloud provider (AWS Lambda, Azure Functions, etc.), making migration difficult. Functions typically have limitations on execution time, memory, and state (they are stateless).

Each level of the Granularity Classification Spectrum serves as a reference point for evaluating and positioning existing microservice architectures. By classifying services along this continuum, architects and development teams can better understand the implications of their granularity choices, iden-

tify potential misalignments, and make informed decisions about boundary adjustments.

3.2.4 Metric Definition

A metric is a quantitative measure of an attribute of an object, system, or process. In software engineering, measurement is essential for planning, controlling, and improving processes, products, and projects Setiadi et al. (2024); Belachew (2018). From this perspective, one of the most important activities in Granulify's granularity mapping is metric collection, which is used to measure granularity saturation (presented in Subsection 3.3).

Although metric definitions are associated with the system context, we suggest a minimum set to capture the consequences of microservice granularity in terms of architectural structure, development effort, deployment stability, code quality, and operational costs. We selected metrics based on three complementary criteria commonly adopted in microservice architecture case studies and modernization: *relevance*, *operational feasibility*, and *comparability*. Below, we present a summary of the metrics and indices used, their purposes, and their connections to the aspects under investigation:

Architectural: Metrics that track the operational state of granularity and cumulative decomposition activity.

- **Number of μServices (NS_t):** number of active microservices deployed in month t, representing the operational state of granularity Bogner et al. (2019); Vera-Rivera et al. (2021);
- **Cumulative Number of μServices (CNS):** total number of services created over time, capturing decomposition activity (including services later consolidated or decommissioned) Abgaz et al. (2023).

Development: Metrics covering team effort and task execution efficiency.

- **Number of Team Members (TM):** indicates the number of people allocated to the modernization effort, capturing the human factor related to adaptation to a given granularity Waseem et al. (2021); Vera-Rivera et al. (2021);
- **Developers per μService (DS):** ratio between team members and the number of services, reflecting the coordination complexity induced by granularity Zhou et al. (2023); Waseem et al. (2021);
- **Estimated Effort (EE):** assesses the predictive capacity of teams given the adopted granularity Hassan et al. (2020); Zhou et al. (2023);
- **Actual Effort (AE):** measures the direct impact of granularity on productivity Becker and Lucrédio (2020); Zhou et al. (2023);
- **Effort Variance (EV):** ratio $\frac{AE}{EE}$, which measures planning efficiency relative to the actual effort required Becker and Lucrédio (2020); Zhou et al. (2023).

Deployment: Metrics that assess stability and change intensity.

- **Number of Changes (NC):** reflects both the volume and intensity of maintenance activities associated with granularity Driessen et al. (2024);
- **Change Frequency (FC):** provides an indication of architectural stability and cohesion over time Driessen et al. (2024); Zhou et al. (2023).

Code Analysis: Metrics that reflect structural complexity and software quality.

- **Code Smells (CS):** counts the number of detected code smells, indicating potential maintainability issues linked to granularity decisions Bogner et al. (2021); Zhong et al. (2022);
- **Thousands of Lines of Code (KLOC):** captures the overall system size and serves as a coarse proxy for codebase complexity in microservice systems Bogner et al. (2019); Driessen et al. (2024);
- **Cyclomatic Complexity (CC):** measures structural complexity as the number of linearly independent control flow paths through the code Bogner et al. (2017); Heitlager et al. (2007);
- **Test Coverage (TC):** reflects the extent to which the codebase is exercised by automated tests, serving as an indicator of testability and confidence against changes Hassan et al. (2020); Taibi and Lenarduzzi (2018).

Operational: Metrics that assess infrastructure costs.

- **Infrastructure Cost (Cost)** is expressed in thousands of US dollars and enables analysis of the financial impact associated with different levels of microservice granularity Hassan et al. (2022); Waseem et al. (2021).

Each metric suggested above is theoretically linked to expected consequences of microservice granularity (e.g., fragmentation influencing architectural size, coordination effort, deployment churn, and operational overhead) and has been repeatedly used in prior empirical work on microservices and software maintainability (as indicated by the studies cited in each dimension).

Furthermore, these metrics can be reliably extracted from existing organizational artifacts (version control, agile tools, CI/CD, and cost dashboards) with limited manual intervention, enabling consistent monthly snapshots and reducing measurement bias.

By grouping the metrics across architectural, development, deployment, code, and operational dimensions, we obtain a balanced view that supports longitudinal comparisons over time, while remaining interpretable and replicable in similar industrial contexts.

3.2.5 Indicators

According to Setiadi et al. (2024), indicators are sets or combinations of metrics transformed into management information, such as quality or process performance indicators.

Based on the metrics presented above, we define four groups of indicators, each synthesizing orthogonal metrics into interpretable signals for granularity management. Each indicator aggregates metrics from a common semantic dimension, eliminating redundancies that would introduce double-counting into the signals. Table 3 presents the four indicators and their component metrics.

Each of these indicators is composed of metrics that, while related, capture distinct aspects of granularity dynamics. The composition is guided by orthogonality principles, ensuring that each indicator represents an independent aspect, avoiding redundancies and multicollinearity that could compromise the interpretation and reliability of the signals. The following paragraphs detail the composition and purpose of each indicator:

I1 — Team Efficiency: This indicator is composed of DS and EV. The composition retains EV—and not EE and AE separately—to avoid double-counting: since $EV = AE/EE$, retaining all three quantities would cause a variation in actual effort to be counted twice within the same indicator, violating the independence assumption in multivariate analysis Hastie et al. (2009). DS captures the coordination load induced by granularity by relating team size to the number of active services. Together, DS and EV signal whether the team is able to absorb and execute the adopted granularity with predictability.

I2 — Delivery Quality: This indicator is composed of CHURN and CFR, complementary metrics that distinguish volume from deployment quality. NC was consolidated into CHURN after verifying that, with fixed monthly periods, $FC \approx NC/30$, generating a correlation > 0.95—a level that characterizes redundancy and introduces multicollinearity Hastie et al. (2009). CFR composes the indicator to capture process reliability beyond volume, measuring the proportion of deployments that resulted in a rollback or a hotfix. Together, CHURN and CFR signal both the pace and stability of the delivery pipeline.

I3 — Code Technical Health: This indicator is composed of TC, CS_DENS, and CC_DENS. The original metrics KLOC, CC, and CS are structurally correlated—larger systems naturally accumulate more cyclomatic complexity and more code smells. To eliminate this dependency, we normalize by size: $CS\_DENS = CS/KLOC$ and $CC\_DENS = CC/KLOC$. Size normalization is established practice in empirical software engineering, enabling fair comparisons across services of different scales Bogner et al. (2017). TC remains as an orthogonal component, capturing confidence against changes independently of size. Together, the three components signal whether the codebase maintains a level of maintainability compatible with the practiced granularity.

I4 — Operational Risk: This indicator is composed of KCOST and HRISK, which cover distinct facets of risk: financial and operational. HRISK was introduced to reveal risk exposure beyond infrastructure cost, measuring the proportion of changes affecting critical system components—a facet that KCOST alone does not capture. Together, KCOST and HRISK signal the cost and fragility associated with the granularity level in operation.

Although the metrics may vary according to organizational context—tool availability, process maturity, and business objectives may lead teams to collect distinct subsets of metrics in each application of the method—indicators I1 through I4 are fixed and stable in Granulify. This distinction is intentional: the indicators represent the invariable analytical dimensions of the method, while the component metrics are the inputs that feed them.

A team may, for example, substitute one effort metric for an equivalent one without altering the structure of the Team Efficiency Indicator (I1), as long as the resulting signal captures the same management facet. This stability ensures that the signals produced by the method are comparable across different industrial contexts and replicable over time.

These four composite indicators, together with the contextual indicators NS and CNS (kept outside the variation model to avoid circular reasoning), form the set of signals used by the Granularity Saturation Method detailed in Section 3.3.

3.2.6 Granularity Mapping Report

After identifying the service topology through granularity level classification and metric collection, the team consolidates the findings into a structured document: the **Granularity Mapping Report**. This report serves as a communication and analysis artifact that synthesizes the findings from the mapping phase, highlighting the distribution of services along the granularity spectrum, identifying architectural risks, and pinpointing critical heterogeneity points. Figure 3 illustrates an example report template, which can be adapted to the specific needs of the industrial context.

After consolidating the **Granularity Mapping Report**, the team analyzes it to extract insights and identify granularity-related "warning signals." These signals include (i) services that do not clearly fit a granularity level, indicating possible ambiguities or misalignments; (ii) coupling or data-sharing patterns that violate data ownership principles, suggesting excessive coupling risks; (iii) services spanning multiple business capabilities without clear justification, indicating under-decomposition; and (iv) excessive fragmentation with many small services, suggesting over-decomposition and potential operational overhead. Identifying these signals is crucial for assessing whether the current granularity is aligned with business needs and for determining whether there is sufficient evidence to justify applying the Granularity Saturation Method in the next phase.

3.3 Measuring Granularity Saturation

This phase applies the *Granularity Saturation Method* (GSM), which consolidates time-series evidence into saturation signals that support adjustment decisions. This method processes the time-series metric and indicator values collected during the Granularity Spectrum Mapping phase to

Table 3. Composite indicators used in the Granularity Saturation Method.

Indicator	Component Metrics	Management Signal
I1 — Team Efficiency	DS	Coordination load induced by granularity; ratio between team members and active services.
	EV	Planning predictability; ratio between actual and estimated effort, signaling whether the team effectively absorbs the adopted granularity.
I2 — Delivery Quality	CHURN	Maintenance intensity; monthly volume of changes capturing the pace of pipeline activity.
	CFR	Delivery reliability; proportion of deployments that required rollback, hotfix, or failure handling.
I3 — Code Technical Health	TC	Confidence against changes; proportion of the codebase covered by automated tests.
	CS_DENS	Normalized maintainability debt; code smells per thousand lines of code.
	CC_DENS	Normalized structural complexity; cyclomatic complexity per thousand lines of code.
I4 — Operational Risk	KCOST	Financial impact; infrastructure cost in thousands of dollars associated with the granularity level.
	HRISK	Operational risk exposure; proportion of changes affecting critical or sensitive components.

quantify how changes in granularity relate to systemic variation across multiple indicators. The method consists of the following steps:

1. **Metric-level variation:** compute the percentage variation of each base metric j in each time period i, applying polarity adjustments and automatic clipping to limit extreme values. This step produces a variation matrix $r_{j,i}$.
2. **Indicator-level aggregation:** aggregate metric-level variations within each indicator d using the median, resulting in indicator-level variations $r_{d,i}$.
3. **Net and Total Variation indices:** aggregate indicator-level variations into two overall indices per period i: *Net Benefit* (NB_i), using the median of signed variations, and *Total Variation* (TV_i), using the median of absolute variations.
4. **Saturation Score and Zone Classification:** compute a bounded *Saturation Score* (SS_i), based on the ratio between NB_i and TV_i, with safeguards against near-zero values. Classify each period into saturation zones (e.g., improvement, churn, deterioration) based on the score and a minimum evidence threshold.

Below we detail each step of the method, illustrating the formulas and design decisions that underpin the granularity saturation quantification approach.

3.3.1 Inputs and Assumptions

Granulify assumes a time series indexed by periods $i \in \{1, 2, \ldots, n\}$ (e.g., weeks, months, quarters, etc.) or equivalent release cycles. For each period i, we observe a set of base metrics indexed by $j \in \{1, 2, \ldots, m\}$, where $x_{j,i}$ denotes the value of metric j in period i.

These metrics are grouped into indicators $d \in \mathcal{D}$ (I1 — Team Efficiency, I2 — Delivery Quality, I3 — Code Technical Health, and I4 — Operational Risk), as defined in Subsection 3.2.5. Each metric j has a polarity $p_j \in \{-1, +1\}$ indicating whether higher values are better ($p_j = +1$) or worse ($p_j = -1$). This separation avoids the semantic incompatibility created when absolute values eliminate directionality between improvements and degradations.

We also track the Cumulative Number of Services (CNS_i) in each period i as a proxy for the granularity level. The method uses the following constants: $c_{clip} = 1.5$ (clipping threshold for percentage variations), $\tau_{zero} = 0.001$ (threshold for near-zero values), $\tau_{tv} = 0.001$ (threshold for near-zero Total Variation), and $c_{eff} = 0.99$ (limit for extreme efficiency values). These values were determined empirically through a pilot analysis on the industrial dataset, to balance outlier resistance with signal preservation Hastie et al. (2009).

The clipping threshold $c_{clip} = 1.5$ allows variations of up to $\pm 150\%$, preventing extreme outliers from dominating subsequent aggregations—a common robustness strategy in time-series analysis Hastie et al. (2009). The near-zero proximity thresholds τ_{zero} and τ_{tv} prevent division-by-zero errors and numerical instability when baseline values or Total Variation approach zero.

The efficiency limit $c_{eff} = 0.99$ bounds extreme ratio values that can arise when comparing planned versus actual effort. Collectively, these parameters provide numerical stability without introducing the sensitivity problems associated with additive epsilon constants in ratio-based calculations, which can artificially inflate percentage variations when denominators are small.

3.3.2 Step 1: Metric-Level Variation

The first step computes the metric-level variation $r_{j,i}$ for each metric j in period i using percentage variation with automatic clipping. For each metric j and period $i \geq 2$, we first compute the raw percentage variation with zero protection:

$$\tilde{r}_{j,i}^{raw} = \begin{cases} 0 & \text{if } |x_{j,i-1}| < \tau_{zero} \\ p_j \cdot \dfrac{x_{j,i} - x_{j,i-1}}{|x_{j,i-1}|} & \text{otherwise} \end{cases} \quad (1)$$

GRANULARITY MAPPING REPORT

TEAM | Custody & Positions Squad **CAPABILITY |** Investment Position Management

DESCRIPTION | Manages the recording, maintenance, and retrieval of client investment holdings and positions across financial instruments.

DECOMPOSED CAPABILITY					DEVELOPMENT			DEPLOYMENT		CODE ANALYSIS				
IMPLEMENTATION UNIT	GCS	URI	BUSINESS SCOPE	FUNCTIONAL SCOPE	EE	AE	EV	NC	FC	CS	KLOC	CC	TC	...
Investment Position Services	**G3**	investment-position-serv	Multiple Capabilities	Services	10	30	3.00	25	0.83	42	1.2	18	89%	...
Retrieval Investment Position	**G5**	retrieval-invest-position	Function	Specific Function	15	34	2.27	60	2.00	28	0.5	8	89%	...
Reconcile Position Changes	**G6**	reconcile-position-changes	Event	Atomic Function	10	29	2.90	34	1.13	14	0.2	4	90%	...
Position Recording	**G4**	position-recording	Sub-capability	Part of the capability	10	20	2.00	30	1.00	35	1.0	15	70%	...

INDICATORS BY QUARTER

QUARTER	CONTEXTUAL		I1 — TEAM EFFICIENCY		I2 — DELIVERY QUALITY		I3 — CODE HEALTH			I4 — OPERATIONAL RISK	
	NS_T	CNS	DS	EV	CHURN	CFR	TC	CS_DENS	CC_DENS	KCOST	HRISK
23Q2	7	15	2.00	3.00	87	5%	85%	2.10	0.20	20	8%
23Q3	15	30	0.80	2.27	**183**	12%	88%	1.80	0.17	20	15%
23Q4	6	36	1.83	2.90	**118**	10%	86%	1.40	0.14	85	12%

RISK SIGNALS

ID	SIGNAL TYPE	AFFECTED UNIT(S)	OBSERVATION	SEVERITY
S1	UNDER-DECOMPOSITION	Investment Position Services (G3)	Business scope is classified as *Multiple Capabilities* at a G3 level, meaning one service absorbs cross-domain responsibilities without bounded context justification. Elevated EV (3.00) and NC (25) reinforce coordination overhead consistent with a service doing too much.	HIGH
S2	GRANULARITY AMBIGUITY	Investment Position Services (G3)	GCS G3 maps to a *single* business capability, yet the Business Scope field reads *Multiple Capabilities*. This conflict indicates either scope creep or misclassification on the spectrum — the service may need re-evaluation against G2 (Service-Based Architecture) criteria before further decomposition decisions.	HIGH
S3	OVER-DECOMPOSITION	Reconcile Position Changes (G6)	Function-level granularity (G6 / FaaS) within a persistent microservice context introduces stateless execution constraints inconsistent with position lifecycle management needs. FC (1.13/month) is elevated for an event-scoped unit, signalling deployment churn typical of premature fragmentation.	MED
S4	COUPLING / DATA SHARING	Position Recording (G4) ↔ Investment Position Services (G3)	Overlapping business scope between these two units creates a shared data dependency risk. The low TC (70%) on Position Recording increases regression exposure whenever cross-service changes propagate through the position lifecycle — a direct consequence of insufficiently separated data ownership.	MED

Figure 3. Example set of decomposition metrics. *Source: the authors' own work.*

The final variation is obtained by applying automatic clipping to limit extreme values:

$$r_{j,i} = \text{clip}(\tilde{r}^{raw}_{j,i}, -c_{clip}, +c_{clip}) \tag{2}$$

The clipping function is defined as $\text{clip}(x, a, b) = \max(a, \min(x, b))$, where the threshold $c_{clip} = 1.5$ allows variations of up to $\pm 150\%$, preventing extreme outliers from dominating subsequent aggregations. This approach is scale-invariant—if all metrics are multiplied by a constant $k > 0$, the percentage variations remain unchanged. A full numerical application using the nine-metric model is provided in the integrated worked example following the regime classification rules.

3.3.3 Step 2: Indicator-Level Aggregation

To obtain an indicator-level view, Granulify aggregates metric-level variations within each indicator d using the **median**:

$$r_{d,i} = \text{median}\left(\{r_{j,i} : j \in \mathcal{M}(d)\}\right) \tag{3}$$

$\mathcal{M}(d)$ denotes the set of metrics belonging to indicator d. The choice of the median (rather than a weighted arithmetic mean) provides robustness against outliers. The median has a breakdown point of 50%, meaning that up to half of the metric values can be arbitrarily extreme without affecting the aggregated result.

In contrast, the arithmetic mean has a breakdown point of 0%—a single outlier can dominate the aggregation. This property is particularly valuable in practice, where individual metrics may exhibit sporadic extreme variations due to data quality issues or exceptional circumstances. The integrated worked example below illustrates this aggregation using the current orthogonal metric set.

3.3.4 Step 3: Net and Total Variation Indices

The third step aggregates indicator-level variations into two overall indices per period i using the **median**. The **Net Benefit** (NB_i) captures the directional balance between improvements and degradations:

$$NB_i = \text{median}\left(\{r_{d,i} : d \in \mathcal{D}\}\right) \tag{4}$$

Simultaneously, we compute the **Total Variation** (TV_i), which quantifies the overall magnitude of change regardless

of direction:

$$TV_i = \text{median}\left(\{|r_{d,i}| : d \in \mathcal{D}\}\right) \tag{5}$$

The use of the median at this aggregation level maintains the robustness properties established in Step 2, ensuring that extreme indicator-level variations do not disproportionately influence the global indices. Positive values of NB_i indicate that improvements dominate degradations, while negative values indicate net deterioration.

TV_i complements this directional signal by measuring the total amount of change observed across indicators. The next step combines these indices into a bounded Saturation Score that enables formal regime classification.

3.3.5 Step 4: Saturation Score and Zone Classification

To quantify whether structural changes translate into net improvements or merely induce churn, we first compute an efficiency ratio $\mathcal{E}_i$ with protection against near-zero Total Variation:

$$\mathcal{E}_i = \begin{cases} 0 & \text{if } TV_i < \tau_{tv} \\ \text{clip}\left(\dfrac{NB_i}{TV_i}, -c_{eff}, +c_{eff}\right) & \text{otherwise} \end{cases} \tag{6}$$

where $\tau_{tv} = 0.001$ is the threshold for near-zero Total Variation and $c_{eff} = 0.99$ prevents extreme efficiency values from dominating the score. The Saturation Score is then computed by normalizing the efficiency ratio to a $[0, 1]$ scale:

$$SS_i = \frac{\mathcal{E}_i + 1}{2} \tag{7}$$

The score SS_i is interpreted as follows: values near 1 indicate that most of the observed change is aligned with improvement ($\mathcal{E}_i \approx +1$), values near 0.5 indicate empirical saturation or balanced churn ($\mathcal{E}_i \approx 0$), and values near 0 indicate net damage ($\mathcal{E}_i \approx -1$).

Since SS_i is ratio-based, its interpretation becomes unreliable when TV_i is very small. Therefore, we classify regimes only when $TV_i \geq \tau_i$, where τ_i is a period-specific minimum evidence threshold estimated from prior observations. When $TV_i < \tau_i$, the period is labeled as *low evidence / stable*, and no saturation regime is assigned. This approach defines saturation as an efficiency ratio between Net Benefit and structural turbulence, capturing whether architectural changes translate into improvement, stagnation, or degradation.

3.4 Lagged Rolling Calibration Protocol and Regime Classification

Rather than computing global thresholds over the entire observation period, Granulify applies a **lagged rolling calibration protocol**. For each period i, the evidence threshold τ_i and saturation bands $q_{25,i}$ and $q_{75,i}$ are computed only from historical periods available before i, bounded by a configurable calibration window. Let w be the maximum number of historical periods used for calibration, and let $W_i = \{k : \max(1, i-w) \leq k < i\}$ be the calibration window for period i. The current period is never included in its own threshold estimate.

Granulify also defines a minimum calibration requirement n_{min}, representing the minimum number of prior observations required before an actionable regime classification can be issued. If $|W_i| < n_{min}$, period i is labeled as having *insufficient calibration history*.

This rule prevents early classifications from being guided by one or two unstable historical points. In operational contexts, both w and n_{min} should be configured according to the review cadence and data availability; in the monthly industrial study reported in this paper, we use $w = 12$ prior monthly observations as the maximum calibration window and $n_{min} = 3$ prior monthly observations as the minimum calibration requirement. The evidence threshold is computed as:

$$\tau_i = Q_{25}(\{TV_k : k \in W_i\}) \tag{8}$$

The saturation bands are then estimated from prior periods with sufficient evidence:

$$V_i = \{SS_k : k \in W_i \wedge TV_k \geq \tau_i\} \tag{9}$$

$$q_{25,i} = Q_{25}(V_i), \qquad q_{75,i} = Q_{75}(V_i) \tag{10}$$

Period i is then classified using τ_i, $q_{25,i}$, and $q_{75,i}$. After classification, period i becomes available to calibrate subsequent periods. Periods labeled as having *insufficient calibration history* are retained for future calibration but are not interpreted as actionable granularity regimes.

This rolling window balances responsiveness to recent architectural changes with sufficient historical depth to produce stable quantile estimates—smaller windows risk volatility caused by single-period outliers, while larger windows dilute sensitivity to emerging trends. We define the following bands:

$$SS_i \leq q_{25,i} \Rightarrow \text{Low (net damage)},$$
$$q_{25,i} < SS_i < q_{75,i} \Rightarrow \text{Neutral (empirical saturation)},$$
$$SS_i \geq q_{75,i} \Rightarrow \text{High (efficient)}.$$

This choice reduces arbitrariness and adapts the thresholds to the observed system dynamics, preserving the semantic interpretation of SS_i around the neutral point 0.5. Figure 4 depicts three scenarios as containers with liquid, where the volume of liquid represents the magnitude of Total Variation (TV_i)—larger volumes indicate more intense architectural change. Color intensity (or shading) reflects the efficiency ratio ($\mathcal{E}_i$), with darker tones indicating negative efficiency (net damage) and lighter or more saturated colors representing positive efficiency (net improvement).

The three zones are visually distinguished: the **Low zone** ($SS_i \leq q_{25,i}$) corresponds to periods in which structural change predominantly causes degradation (net damage), often visualized with darker, turbulent liquid; the **Neutral zone** ($q_{25,i} < SS_i < q_{75,i}$) represents empirical saturation, where change produces neither consistent improvement nor damage (balanced churn); and the **High zone** ($SS_i \geq q_{75,i}$) indicates efficient change, where the observed variation translates into net architectural improvement. The integrated worked example below operationalizes this visual metaphor using the time

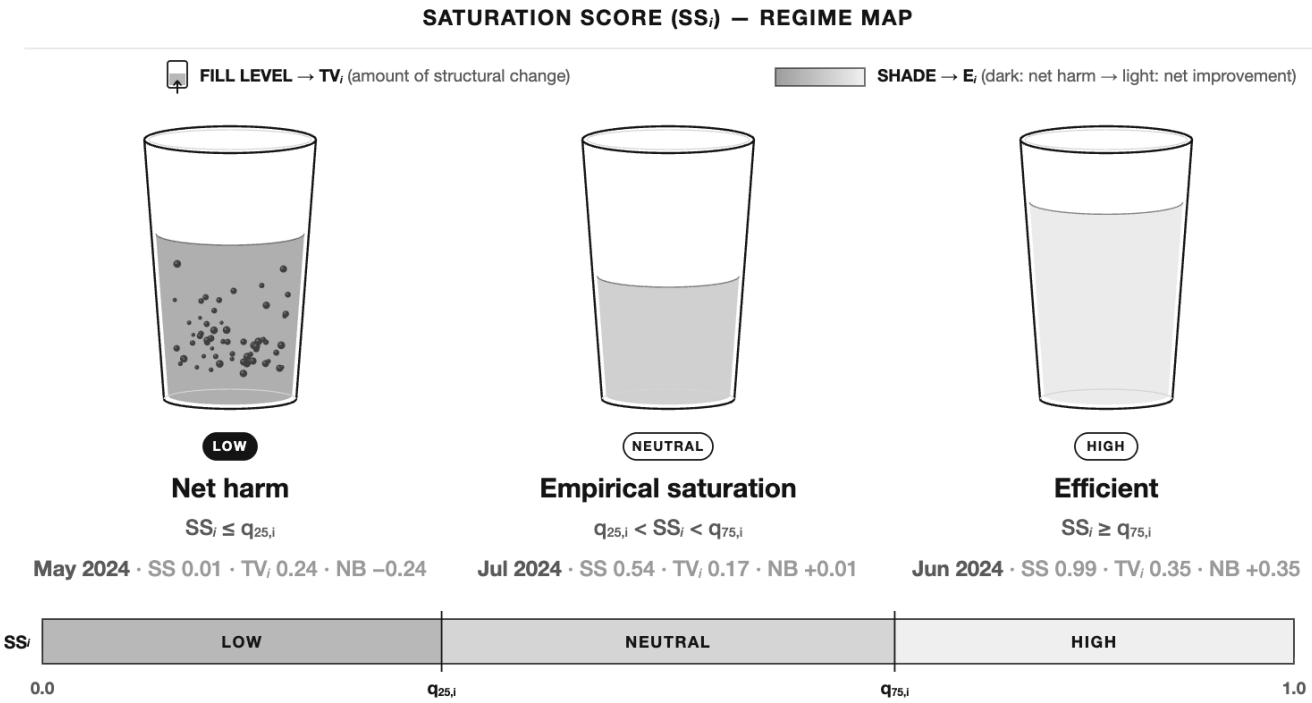

Figure 4. Illustration of empirical saturation zones based on the Saturation Score (SS). The volume of liquid represents Total Variation (TV), while color intensity reflects efficiency. Rolling quantile-based thresholds ($q_{25,i}$, $q_{75,i}$) define the formal score bands used for regime classification.

series extracted from the industrial study to illustrate regime classification in a real-world scenario.

Given the granularity proxy CNS_i (number of services in period i), we classify the system into qualitative regimes based on the trend of CNS and the calibrated bands of SS, subject to the evidence floor $TV_i \geq \tau_i$.

Table 4 summarizes the classification rules and associated advisory actions. These rules combine the availability of historical calibration data, the direction of granularity change ($CNS_i \uparrow$, $CNS_i \downarrow$, or $CNS_i =$), and the Saturation Score bands (high, neutral, low) to produce the regime states used by Granulify.

3.5 Integrated Worked Example

To make the calculation concrete, consider the transition from May 2024 to June 2024 in the industrial time series reported in Table 6. The cumulative number of services increased from $CNS = 54$ to $CNS = 56$, so the period is evaluated as a decomposition event.

For metrics whose increase is undesirable ($p_j = -1$), such as DS, EV, CHURN, CFR, CS_DENS, CC_DENS, KCOST, and HRISK, decreases produce positive variation; for TC ($p_j = +1$), increases produce positive variation. For example, the improvement in effort variance is computed as:

$$r_{\text{EV,Jun}} = -1 \cdot \frac{1.944 - 3.520}{3.520} = +0.448 \qquad (11)$$

Applying the same polarity-adjusted calculation to the nine metrics for June 2024 yields:

$$\begin{aligned} r_{\text{DS}} &= +0.028, & r_{\text{EV}} &= +0.448, \\ r_{\text{CHURN}} &= +0.700, & r_{\text{CFR}} &= +0.584, \\ r_{\text{TC}} &= -0.022, & r_{\text{CS\_DENS}} &= -0.060, \\ r_{\text{CC\_DENS}} &= +0.065, & r_{\text{KCOST}} &= -0.063, \\ r_{\text{HRISK}} &= +1.000. \end{aligned}$$

Step 2 aggregates these metric-level variations within each indicator using the median, producing the indicator-level variations reported in Table 6:

$$\begin{aligned} r_{\text{I1}} &= \text{median}(0.028, 0.448) \approx 0.238, \\ r_{\text{I2}} &= \text{median}(0.700, 0.584) \approx 0.641, \\ r_{\text{I3}} &= \text{median}(-0.022, -0.060, 0.065) \approx -0.022, \\ r_{\text{I4}} &= \text{median}(-0.063, 1.000) \approx 0.468. \end{aligned}$$

Steps 3 and 4 then aggregate the per-indicator variations into NB, TV, and SS:

$$\begin{aligned} NB_{\text{Jun}} &= \text{median}(0.238, 0.641, -0.022, 0.468) \\ &= \frac{0.238 + 0.468}{2} \approx 0.35, \\ TV_{\text{Jun}} &= \text{median}(0.238, 0.641, 0.022, 0.468) \\ &= \frac{0.238 + 0.468}{2} \approx 0.35, \\ \mathcal{E}_{\text{Jun}} &= \text{clip}\left(\frac{0.35}{0.35}, -0.99, +0.99\right) = 0.99, \\ SS_{\text{Jun}} &= \frac{0.99 + 1}{2} = 0.995 \approx 0.99. \end{aligned}$$

For June 2024, the rolling calibration values are $\tau_i = 0.12$, $q_{25,i} = 0.14$, and $q_{75,i} = 0.56$. Since $TV_i = 0.35 \geq \tau_i$, $SS_i = 0.99 \geq q_{75,i}$, and CNS_i increased, Table 4 classifies

Table 4. Instantaneous regime classification and advisory actions per period using the Saturation Score SS_i, with arrows indicating increase (↑) or decrease (↓) in granularity (CNS_i), subject to the period-specific evidence floor ($TV_i \geq \tau_i$).

Observed Condition	Regime Interpretation	Advisory Action
Insufficient prior observations in W_i	**Insufficient Calibration History:** The period contributes to future calibration, but no actionable regime is assigned.	Accumulate historical observations; defer structural intervention until sufficient calibration history is available.
$TV_i < \tau_i$	**Low Evidence / Stable:** The observed structural variation is insufficient to sustain a reliable ratio-based interpretation; no saturation regime is assigned.	Maintain current boundaries and reassess in the next cycle.
$CNS_i =$	**No Change:** No change in the number of services occurred in the current period; dimensional variation is monitored, but no splitting/merging regime is assigned.	Continue monitoring dimensional variation; no splitting or merging action is indicated.
$CNS_i \uparrow, SS_i \geq q_{75,i}$	**Efficient Decomposition:** A granularity increase in the current period is associated with predominantly positive Net Benefit per unit of structural change.	Continue the current decomposition strategy; no immediate intervention is required.
$CNS_i \uparrow, q_{25,i} < SS_i < q_{75,i}$	**Empirical Saturation (Plateau):** Granularity continues to increase, but no observable Net Benefit is achieved relative to the induced change intensity.	Pause further fragmentation; investigate whether pending splits deliver clear bounded value before proceeding.
$CNS_i \uparrow, SS_i \leq q_{25,i}$	**Over-Decomposition (Net Damage):** A granularity increase in the current period is associated with net degradation, suggesting possible over-decomposition.	Evaluate service merges; prioritize candidates with high coupling or shared data access across boundaries.
$CNS_i \downarrow, SS_i \geq q_{75,i}$	**Efficient Consolidation:** A granularity reduction is associated with positive Net Benefit per unit of structural change.	Continue consolidation; monitor to avoid under-decomposition.
$CNS_i \downarrow, SS_i \leq q_{25,i}$	**Harmful Consolidation:** Consolidation in the current period is associated with net degradation.	Stop consolidation; review whether merged boundaries are causing bottlenecks or increased coordination cost.

the period as *Efficient Decomposition*. The same calculation pipeline produces different regime interpretations across the longitudinal sequence, as summarized in Table 5.

The example shows how Granulify separates three concerns that are frequently conflated in granularity decisions: the amount of observed change (TV), the direction of its systemic effect (NB and SS), and the architectural meaning of that effect given the service count trend (CNS).

This is why a period with increasing granularity can indicate either efficient decomposition or over-decomposition, depending on whether the induced variation translates into positive Net Benefit.

The final output of the Granularity Saturation Method is the **Granularity Saturation Signal**, a time series of regime classifications for each observation period i. For each period, the signal assigns one of the regime values detailed in Table 4, which are determined by the availability of historical calibration data, the Saturation Score zone (SS_i relative to $q_{25,i}$ and $q_{75,i}$), the evidence floor τ_i, and the direction of granularity change ($CNS_i \uparrow$, $CNS_i \downarrow$, or $CNS_i =$). The same table also reports the advisory action associated with each regime.

The signal structure enables longitudinal analysis by tracking regime transitions over time. For example, a sequence *Efficient Decomposition* → *Empirical Saturation* → *Over-Decomposition* would indicate progressive degradation as fragmentation increases, signaling the need to stop further decompositions or initiate consolidation. Conversely, sustained *Efficient Decomposition* regimes confirm that granularity adjustments continue to deliver net architectural benefits.

This temporal regime classification serves as a diagnostic tool for informing architectural decision-making, indicating whether current granularity adjustments are producing beneficial outcomes or whether saturation effects are manifesting.

3.6 Decision Support

Granulify was designed as a **decision support tool**, not as an automation mechanism. Splitting, merging, and reorganization decisions remain the responsibility of the architecture team, which contextualizes the method's quantitative signals with organizational constraints, such as team topology, business priorities, deployment risk, and migration roadmap.

The saturation signal is reviewed in monthly architecture meetings, where the team interprets the regime classification together with qualitative observations from sprint retrospectives before committing to any structural change. When a

Table 5. Integrated worked example of regime classification across selected periods of the industrial time series. *Source: the authors' own work.*

Period	CNS	NB/TV/SS	Band Test	Regime Interpretation
Feb 2024	↑	-0.12 / 0.12 / 0.01	$TV_i \geq 0.11$; $SS_i \leq 0.24$	*Over-decomposition*: produced net damage.
…	…	…	…	…
Jun 2024	↑	0.35 / 0.35 / 0.99	$TV_i \geq 0.12$; $SS_i \geq 0.56$	*Efficient decomposition*: produced positive net benefit.
Jul 2024	↑	0.01 / 0.17 / 0.54	$TV_i \geq 0.12$; $0.24 < SS_i < 0.78$	*Empirical saturation*: yielded limited net benefit.
…	…	…	…	…
Sep 2024	↓	0.37 / 0.37 / 0.99	$TV_i \geq 0.14$; $SS_i \geq 0.67$	*Efficient consolidation*: improved the system.

regime justifies action, the decision and its rationale are documented in an Architecture Decision Record (ADR), closing the feedback loop with the macroprocess described in Section 3.1.

Each regime carries an implicit architectural recommendation, as summarized in Table 4. These recommendations are advisory: the team may choose to defer action based on context (e.g., an ongoing production freeze, team capacity constraints), but the regime provides a structured starting point for discussion.

When Net Benefit (NB_i) is negative, the per-indicator variations—r_{I1} (I1 — Team Efficiency), r_{I2} (I2 — Delivery Quality), r_{I3} (I3 — Code Technical Health), and r_{I4} (I4 — Operational Risk)—identify which indicator drove the degradation, enabling root-cause reasoning rather than treating all negative signals uniformly:

- A strongly negative r_{I1} (**I1 — Team Efficiency**) signals planning inefficiency: actual effort is diverging significantly from estimates, often a symptom of unclear service boundaries leading to coordination overhead across teams. The team should review whether Effort Variance (EV) and Developer Density (DS) point to services that require disproportionate maintenance relative to their scope.
- A strongly negative r_{I2} (**I2 — Delivery Quality**) with an overall negative NB suggests that the current granularity level is producing deployment instability—artifacts too independent to coordinate safely. The recommended focus is on reducing change failure rates (CFR) and change volume (CHURN), which may indicate premature decomposition of services that remain strongly coupled at the data or API layer.
- A strongly negative r_{I3} (**I3 — Code Technical Health**), in isolation, is typically less actionable at the boundary level, as it frequently reflects accumulated technical debt within services rather than boundary misalignment. However, when combined with negative r_{I1} or r_{I2}, it may indicate services whose scope has grown too broad.
- A strongly negative r_{I4} (**I4 — Operational Risk**) indicates that infrastructure costs or high-risk change rates are scaling beyond the baseline trend. This may reflect over-provisioned services or services whose deployment frequency is causing elevated incident exposure.

When multiple indicators are simultaneously negative, the team prioritizes the indicator with the largest absolute variation, as it represents the strongest systemic signal in that observation cycle. In cases where indicators conflict—one positive, another negative—the overall NB signal governs whether intervention is warranted, while the per-indicator breakdown informs which aspect of the architecture to address first.

The next section presents the research method and industrial study design that grounded this approach.

4 Research Method

In the previous section, we presented Granulify as a prescriptive method for continuous microservice granularity management, detailing its central artifacts and decision-making process. In this section, we describe the research methodology that underlies the development and evaluation of Granulify in a real industrial context.

This study followed **Design Science Research** (DSR) as an industry-oriented methodological approach for designing, refining, and evaluating *Granulify*. We consider DSR appropriate because our goal is not merely to explain a phenomenon, but also to construct an actionable solution and evaluate its utility in a real context Dresch et al. (2020).

We operationalized DSR through **Action Research** Avison et al. (1999) conducted in a large-scale industrial platform. Action research is appropriate when the researcher actively participates in organizational change while systematically studying the intervention process Dresch et al. (2020).

In our context, the first author assumed the role of lead architect, enabling both active contribution to granularity decisions and systematic observation of how the Granulify approach influenced architectural evolution in practice.

In the following subsections, we detail the guiding hypotheses that drive the empirical investigation and the study design, including the definition of the unit of analysis, the observation period, the participants and roles involved, the data sources, and the triangulation procedure to mitigate biases and strengthen the validity of the results.

4.1 Guiding Hypotheses

In order to guide the empirical investigation, we formulated two hypotheses that reflect the underlying assumptions about the need for a systematic approach to microservice granularity management:

- **HB$_1$ (Diagnostic):** In the absence of systematic granularity management mechanisms, microservice boundaries tend to evolve in an *ad hoc* manner, which likely

degrades architectural quality and increases maintenance and operational load over time.
- **HB_2 (Intervention):** The introduction of systematic, evidence-based practices for assessing and adjusting granularity improves architectural quality and maintainability, by enabling more informed splitting/merging decisions.

These hypotheses are explored through longitudinal indicators already available in the organization, covering service count dynamics (e.g., NS_t), delivery effort and planning stability (e.g., AE and EV), change intensity and code characteristics (e.g., NC, FC, KLOC, and CC), quality and risk-reduction signals (e.g., TC), and infrastructure cost (Cost). Furthermore, we analyze granularity saturation signals using the dedicated Granularity Saturation Method (GSM) to identify variation and magnitude patterns as granularity evolves.

4.2 Study Design

The research was conducted in a large-scale investment management platform belonging to one of Brazil's largest private financial institutions. The platform manages more than 40 million monthly transactions and serves more than 250,000 internal users in the investment, compliance, and trading domains.

This context was particularly suitable for the development and evaluation of Granulify due to the ongoing monolith-to-microservices migration and the recurring challenges faced by teams in defining and revising service boundaries over time.

We executed the research in iterative cycles in which (i) Granulify's artifacts and decision criteria were refined, (ii) monthly indicators were collected from organizational repositories, and (iii) the results were interpreted with practitioner participation to support boundary assessment and adjustment.

To avoid temporal leakage, regime classification followed the lagged rolling protocol defined in 3.3.5, such that each monthly observation was classified only with thresholds calibrated from prior observations.

4.2.1 Unit of Analysis

The primary unit of analysis is the evolution of the platform's microservice architecture during a modernization initiative, with emphasis on granularity-related decision-making and its observable effects across multiple dimensions (architectural structure, development effort, operational change, code characteristics, and infrastructure cost).

Concretely, we focus on three groups of critical capabilities migrated by the observed team: *Financial Instruments Management*, *Investment Positions Management*, and *Investment Portfolio Management* [6].

These capabilities were progressively decomposed into microservices as part of the institution's ongoing modernization initiative, enabling longitudinal observation of fragmentation and consolidation patterns under real organizational constraints.

Originally designed to support a limited daily transaction load, the institution's *Asset Management System* evolved to meet the growing demands of the investment sector. Initially monolithic, the architecture was progressively fragmented to meet scalability, maintainability, robustness, and business alignment requirements.

Throughout this process, the team faced the practical reality that there are few established patterns for determining how granular a microservice should be. The resulting challenges, related to productivity and operational costs, make this trajectory particularly suitable for motivating and grounding the proposed continuous granularity management approach.

4.2.2 Observation Period

The analyzed period spans from September 2023 to September 2024, during which the institution executed a new migration phase to further modernize the system. This interval enables us to capture architectural changes before and after the system entered sustained production use, allowing temporal comparison across monthly observation cycles.

The monthly observation frequency was defined by this study to balance temporal granularity with data collection feasibility. Monthly snapshots provide sufficient resolution to capture architectural evolution patterns and team adaptation dynamics, while avoiding excessive measurement overhead.

This cadence proved adequate for observing significant changes in service granularity, development effort, and operational metrics following architectural decisions.

4.2.3 Participants and Roles

The modernization effort involved 3 cross-functional teams, each composed of 5 members, including software engineers, quality assurance specialists, and a technical lead. Collectively, approximately 15 professionals contributed to the platform's evolution during the observed period, with team size varying from 10 to 15 members depending on migration capacity.

The first author served as the lead architect responsible for granularity decisions, having direct access to architectural decision-making processes, sprint planning sessions, and retrospective meetings. This role enabled continuous observation of granularity-related challenges, systematic collection of engineering metrics, and active participation in design discussions. The embedded position facilitated data triangulation, combining participant observation with access to the quantitative and qualitative sources described below.

Qualitative interpretation was based on architecture review discussions that were already part of the team's normal governance practice, rather than on interviews created specifically for the study. These discussions involved the lead architect, technical leads, and senior team members, and were reviewed quarterly using the monthly checkpoints as input.

The quantitative signals produced by Granulify were used to challenge and refine the team's existing interpretation of

[6] The original business capability names were replaced with terms from the BIAN (Banking Industry Architecture Network) model—`https://bian.org`

architectural conditions, rather than to replace practitioner judgment. When quantitative signals and team perception diverged, the team performed root-cause analysis, created investigation items, and validated the interpretation with technical leads before using the evidence to support boundary decisions.

4.2.4 Data Sources

Metrics were collected through dedicated organizational tooling across all five measurement dimensions. Code quality metrics (TC, CS, CC, KLOC) were obtained from SonarQube, which was integrated into the organization's CI/CD pipeline and configured to run for each pull request and as part of nightly builds. Deployment metrics (CHURN, CFR) were derived from GitHub Actions execution logs combined with AWS CloudWatch event streams, which captured all deployment events and failure incidents. Development metrics (DS, EV) were extracted from ServiceNow, the organization's service management platform, using sprint-level records of estimated and actual working days per team. Infrastructure cost metrics (KCOST, HRISK) were obtained from AWS Cost Explorer using service-level cost allocation tags maintained in the cloud billing system. Architectural metrics (NS, CNS) were tracked manually from service inventories updated at each architecture review meeting.

Qualitative evidence was collected from governance artifacts already used by the team, including architecture review discussions, revised architecture diagrams, Architecture Decision Records (ADRs), technical debt records, and backlog decisions. These qualitative sources were not used to compute the quantitative indices (NB, TV, or SS). Instead, they supported the interpretation of regime classifications, helped challenge quantitative signals against practitioner perception, contextualized splitting/merging/consolidation decisions, and provided traceability for actions resulting from review discussions.

4.2.5 Measurement Pipeline

Raw metric data were collected through the public APIs of each source tool at the end of each monthly observation cycle. The extracted datasets were ingested into an AWS Athena analytical environment, where they were partitioned by sprint date and by project, establishing a single source of truth for all quantitative analyses reported in this paper and enabling consistent queries across periods without data leakage between observation windows.

Before analysis, each monthly snapshot underwent a two-step normalization procedure: (*i*) structural normalization, in which size-dependent metrics (CS, CC, KLOC) were converted to density measures (CS_DENS, CC_DENS), enabling fair comparison across periods regardless of system growth; and (*ii*) temporal alignment, in which ServiceNow sprint boundary timestamps were used to align effort and change records to the corresponding calendar month, resolving cases where sprints crossed month boundaries.

Records with missing values (occurring in three isolated instances due to tooling configuration gaps at the beginning of the observation period) were excluded from the dimensional classification calculation for that period, rather than imputed, preserving measurement integrity. The complete extraction, integration, and normalization cycle required approximately 8 hours of senior engineering effort per month.

4.2.6 Triangulation Procedure

Here the term "triangulation" is used to describe the process of integrating and challenging multiple evidence sources (quantitative and qualitative) to strengthen the validity of granularity-related interpretations and decisions.

This process followed a recurring review protocol aligned with the monthly measurement cycle. First, Granulify produced quantitative signals from the monthly snapshot, including dimensional classifications, Net Benefit (NB), Total Variation (TV), Saturation Score (SS), and the resulting regime classification.

These signals were computed before qualitative interpretation and were not manually adjusted during the review. Second, the monthly signals were presented at architecture review checkpoints and consolidated quarterly with the lead architect, technical leads, and senior team members.

In these reviews, the team compared three layers of evidence: (*i*) the quantitative behavior observed in Granulify's indicators; (*ii*) the practitioners' interpretation from architecture review discussions; and (*iii*) the governance artifacts described in the data sources.

Agreement across these layers strengthened the interpretation of a regime as an actionable architectural condition. Divergence between quantitative signals and team perception triggered root-cause analysis, creation of an investigation item, and validation with technical leads before the interpretation was used to support a boundary decision.

4.2.7 Bias Mitigation

We mitigated biases through (i) triangulation across independent organizational sources (engineering, delivery, operations, and finance); (ii) consistent monthly snapshots over a pre-defined time interval; and (iii) involvement of multiple technical roles in qualitative classification and interpretation activities to reduce single-observer bias. Additionally, since metric extraction relied on existing organizational APIs and dashboards rather than manual observation, the risk of selective data capture was structurally reduced.

5 Results and Discussion

This section presents the results of applying the Granulify method in the industrial case study described in Subsection 4.2. The analysis is structured in three main parts: (*i*) an overview of the collected data and the observation period; (*ii*) a detailed analysis of the granularity saturation signals identified over time using the method; and (*iii*) an evaluation of the diagnostic and intervention hypotheses formulated in Section 4 in light of the observed results. Finally, we discuss the practical implications of these results for granularity management in microservice systems and the lessons learned for future research and applications of the Granulify method.

5.1 Data Overview and Observation Period

The application of the method can be understood in two distinct phases, each with specific characteristics and objectives. The initial **retrospective analysis** phase (September 2023–February 2024) focused on collecting and analyzing historical data already available through the existing observability infrastructure. This phase enabled us to establish a baseline for the architectural, development, deployment, code analysis, and operational metrics, as well as for the composite indicators used in the variation analysis.

During this phase, the system was in a pre-production state, meaning that teams had more freedom to experiment with granularity adjustments without the operational constraints typical of a production environment. This resulted in a higher frequency of granularity changes, reflected in more volatile variations in the composite indicators.

In the subsequent **prospective continuous monitoring** phase (March 2024–September 2024), the focus shifted to observing the effects of granularity changes in real time, as the system evolved and approached production. In this phase, teams began to adopt a more structured approach to granularity management, using the signals generated by the Granulify method to guide their decisions.

This timeline allowed us to compare pre-production patterns—characterized by frequent exploratory decomposition—with production-constrained patterns, in which consolidation and stability concerns became more prominent.

Figure 5 summarizes the observation timeline, highlighting the retrospective and prospective phases, the production transition, the monthly regime classifications, and the main architectural decision points discussed in the architecture review checkpoints.

Table 6 presents the monthly values of the collected metrics, together with the indicator-level variations computed from these metrics. The metrics are organized across the architectural (NS, CNS), development (DS, EV), deployment (CHURN, CFR), code analysis (TC, CS_DENS, CC_DENS), and operational (KCOST, HRISK) dimensions.

Each indicator-level variation column (r_{I1}, r_{I2}, r_{I3}, and r_{I4}) is positioned within the dimension of its respective composite indicator—I1 — Team Efficiency, I2 — Delivery Quality, I3 — Code Technical Health, and I4 — Operational Risk, as defined in Section 3.2.5—making explicit the link between the collected metrics and the composite management signals.

The architectural metrics (NS and CNS) serve as contextual proxies for the granularity level, but do not form part of any indicator and are not included in the variation analysis to avoid circular reasoning (using the granularity change to predict itself). The first monthly snapshot is used as the reference baseline and therefore has no indicator-level variation computed.

5.2 Signal Processing and Granularity Saturation Analysis

As defined in 3.2.5, Granulify organizes metrics into four composite indicators used for variation analysis: I1 — Team Efficiency (*Development*), I2 — Delivery Quality (*Deployment*), I3 — Code Technical Health (*Code Analysis*), and I4 — Operational Risk (*Operational*).

The architectural metrics (NS and CNS) are excluded from the variation analysis to avoid circular reasoning—using the granularity change (CNS) to predict the effects of the granularity change would create a tautological measurement artifact.

For each monthly observation period after the September 2023 baseline, we computed the indicator-level variations r_{I1} (I1), r_{I2} (I2), r_{I3} (I3), and r_{I4} (I4), applying the metric-level variation and indicator-level aggregation steps described in 3.3.2 and 3.3.3. These variations are reported alongside their component metrics in Table 6.

After obtaining the indicator-level variations, we computed the Net Benefit (NB) and Total Variation (TV) indices for each period using the methodology described in 3.3.4 and 3.3.5. From these indices, we derived the Saturation Score (SS) for each observation period.

Regime classification followed the lagged rolling calibration protocol defined in 3.3.5. The main configuration used $w = 12$ and $n_{min} = 3$, such that the initial months contributed to calibration but were not interpreted as actionable granularity regimes.

Table 7 consolidates the observed indices, the period-specific calibration thresholds used in the main configuration, and the regime classifications obtained under alternative calibration windows. The $w = 12$ column represents the main interpretation used in this study, while $w = 3$ and $w = 6$ are sensitivity checks with shorter historical horizons.

The shorter windows are not competing final results; they indicate how the diagnosis would change if the method prioritized more recent observations. In practical terms, $w = 3$ represents a more reactive, but less stable, calibration; $w = 6$ provides an intermediate horizon; and $w = 12$ smooths short-term fluctuations by using the longest monthly history available in this dataset.

This sensitivity comparison helps assess whether the observed regimes depend excessively on a single calibration choice. Agreement across $w = 3$, $w = 6$, and $w = 12$ indicates more robust regime signals, while disagreement indicates that the diagnosis is sensitive to the calibration horizon and should be interpreted conservatively.

For example, the windows agree on *Low evidence/stable* for January, *Efficient decomposition* for April and June, and *Efficient consolidation* for September. In contrast, February, July, and August show disagreement across windows, indicating that the over-decomposition and empirical saturation signals in those months should be interpreted cautiously, as they depend on the calibration history considered.

These stable classifications provide stronger evidence that the corresponding architectural events are not artifacts of the selected $w = 12$ window. The main sensitivity appears at the degradation/saturation boundary: with $w = 6$, August 2024 becomes *Over-decomposition* instead of *Empirical saturation*; with $w = 3$, February, July, and August are treated as *Low evidence/stable*. Therefore, we conservatively interpret July–August 2024 as a saturation/degradation warning region, treating the exact boundary between empirical saturation and over-decomposition as window-sensitive.

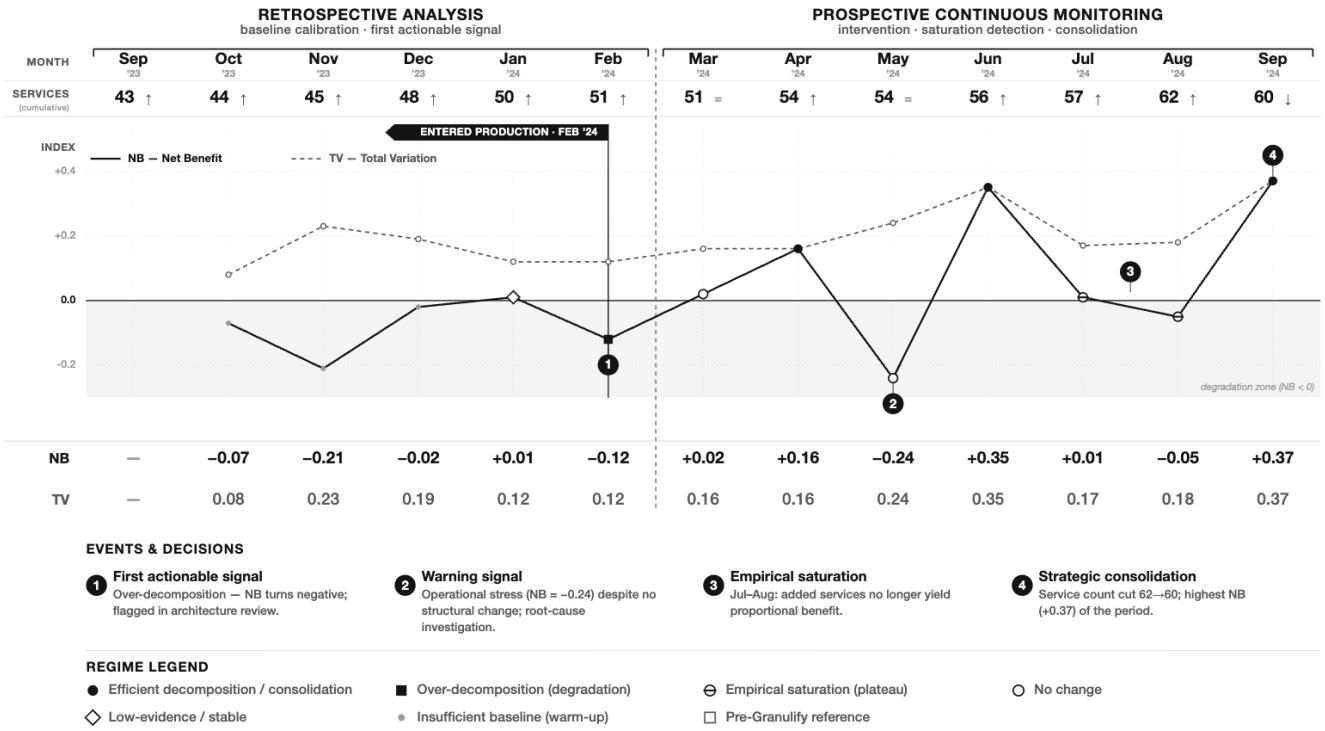

Figure 5. Longitudinal timeline of the Granulify observation period, showing the retrospective analysis phase, the prospective continuous monitoring phase, the production transition, the monthly regime classifications, and the main architectural decision points. *Source: the authors' own work.*

5.3 Diagnostic Hypothesis Evaluation

In this section, we evaluate **HB_1 (Diagnostic)** in light of the results of the granularity saturation analysis. HB_1 proposes that, in the absence of systematic granularity management mechanisms, microservice boundaries evolve in an *ad hoc* manner, producing degradation signals in the number of services, planning stability, deployment activity, code characteristics, and operational cost.

To evaluate this hypothesis, we examined the variation patterns in the composite indicators and regime classifications throughout the retrospective period, seeking degradation signals associated with ad hoc granularity evolution. This **retrospective** analysis revealed that between September 2023 and February 2024 the system underwent several granularity adjustments without a structured approach:

- **September 2023:** with 43 cumulative microservices (as shown in Table 6), the system was in a pre-Granulify state, serving as the initial reference point for subsequent variation analysis;
- **October–December 2023:** these months showed negative and unstable variation patterns, including elevated planning inefficiency and code smell density, but the rolling calibration protocol classified them as having *Insufficient calibration history* because fewer than $n_{min} = 3$ prior monthly variation observations were available. Therefore, they provided data for subsequent calibration rather than supporting actionable regime claims;
- **January 2024:** the system was classified as *Low evidence/stable*, with a slightly positive Net Benefit (NB = 0.01), Total Variation of 0.12, and Saturation Score of 0.55, suggesting that the structural variation was not strong enough to sustain an actionable saturation regime;
- **February 2024:** the first actionable degradation signal emerged when the system was classified as *Over-decomposition*. This period combined an increase in cumulative services (CNS = 51), negative Net Benefit (NB = -0.12), low Saturation Score (SS = 0.01), the highest change volume in the retrospective period (CHURN = 51), low planning efficiency (EV = 3.483), a peak infrastructure cost of $45k, and the first occurrence of high-risk changes (HRISK = 3.85). Together, these signals indicate that continued fragmentation was no longer producing proportional architectural benefit.

The saturation signals identified during this retrospective analysis were validated against the qualitative evidence described in Section 4.2.6. For February 2024, the *Over-decomposition* regime coincided with practitioner concerns about deployment pressure, increased coordination overhead, and escalating costs, supporting the interpretation that the architecture had reached a degradation point associated with ad hoc decomposition.

This supports HB_1 by indicating that systematic granularity mechanisms can detect degradation patterns once sufficient historical evidence has been accumulated, avoiding the post-hoc interpretation of early periods that lack sufficient calibration history.

5.4 Intervention Hypothesis Evaluation

In this section, we evaluate **HB_2 (Intervention)** in light of the results of the granularity saturation analysis and the associated qualitative evidence. HB_2 proposes that the introduction of a systematic granularity management method, such

Table 6. Collected metrics and indicator-level variations over time. *Source: the authors' own work.*

Architectural		Development			Deployment			Code Analysis				Operational		
Period	**CNS**	**DS**	**EV**	r_{I1}	**CHURN**	**CFR**	r_{I2}	**TC**	**CS_DENS**	**CC_DENS**	r_{I3}	**KCOST**	**HRISK**	r_{I4}
2023-09	43 ↑	0.391	2.142	–	22	0.00	–	89%	8.811	0.486	–	18	0.00	–
2023-10	44 ↑	0.375	2.889	-0.153	28	3.12	-0.136	89%	9.325	0.450	-0.001	17	0.00	0.027
2023-11	45 ↑	0.375	2.645	0.042	41	7.69	-0.964	88%	12.738	0.405	-0.003	31	0.00	-0.411
2023-12	48 ↑	0.346	4.541	-0.319	14	0.00	0.829	90%	10.830	0.404	0.016	35	0.00	-0.064
2024-01	50 ↑	0.333	2.873	0.202	21	4.76	-0.250	92%	7.551	0.388	0.039	36	0.00	-0.014
2024-02	51 ↑	0.333	3.483	-0.106	51	3.85	-0.618	90%	7.320	0.380	0.020	45	3.85	-0.125
In production														
2024-03	51 =	0.333	3.195	0.041	39	7.89	-0.407	90%	7.615	0.365	-0.003	34	2.63	0.280
2024-04	54 ↑	0.300	2.457	0.165	24	8.33	0.164	90%	8.019	0.352	0.002	40	0.00	0.411
2024-05	54 =	0.281	3.520	-0.184	40	20.00	-1.033	90%	7.864	0.339	0.019	63	2.50	-0.287
2024-06	56 ↑	0.273	1.944	0.238	12	8.33	0.641	88%	8.333	0.317	-0.022	67	0.00	0.468
2024-07	57 ↑	0.303	2.972	-0.319	12	0.00	0.500	89%	8.723	0.308	0.011	65	30.77	0.014
2024-08	62 ↑	0.303	3.598	-0.105	06	0.00	0.250	88%	8.739	0.304	-0.001	56	60.00	-0.405
2024-09	60 ↓	0.303	1.303	0.319	01	5.33	0.417	90%	8.076	0.303	0.020	50	8.31	0.484

Table 7. Granularity regime classification, rolling calibration thresholds, and window sensitivity. *Source: the authors' own work.*

Period	**CNS**	**NB**	**TV**	**SS**	$\lvert W_i \rvert$	τ_i	$q_{25,i}$	$q_{75,i}$	$w = 3$ **(check)**	$w = 6$ **(check)**	$w = 12$ **(main)**
2023-10	44 ↑	-0.07	0.08	0.08	00	–	–	–	Insuf. calib. history	Insuf. calib. history	Insuf. calib. history
2023-11	45 ↑	-0.21	0.23	0.04	01	–	–	–	Insuf. calib. history	Insuf. calib. history	Insuf. calib. history
2023-12	48 ↑	-0.02	0.19	0.44	02	–	–	–	Insuf. calib. history	Insuf. calib. history	Insuf. calib. history
2024-01	50 ↑	0.01	0.12	0.55	03	0.14	0.14	0.34	Low evid./stable	Low evid./stable	Low evid./stable
2024-02	51 ↑	-0.12	0.12	0.01	04	0.11	0.24	0.49	Low evid./stable	Over-decomposition	Over-decomposition
In production											
2024-03	51 =	0.02	0.16	0.56	05	0.12	0.03	0.47	No change	No change	No change
2024-04	54 ↑	0.16	0.16	0.99	06	0.12	0.34	0.55	Efficient decomp.	Efficient decomp.	Efficient decomp.
2024-05	54 =	-0.24	0.24	0.01	07	0.12	0.44	0.56	No change	No change	No change
2024-06	56 ↑	0.35	0.35	0.99	08	0.12	0.14	0.56	Efficient decomp.	Efficient decomp.	Efficient decomp.
2024-07	57 ↑	0.01	0.17	0.54	09	0.12	0.24	0.78	Low evid./stable	Empirical sat.	Empirical sat.
2024-08	62 ↑	-0.05	0.18	0.35	10	0.13	0.24	0.78	Low evid./stable	Over-decomposition	Empirical sat.
2024-09	60 ↓	0.37	0.37	0.99	11	0.14	0.27	0.67	Efficient consol.	Efficient consol.	Efficient consol.

as Granulify, can support more informed splitting/merging decisions, helping architecture teams to reason about architectural quality and to identify signals of efficient decomposition, operational stress, empirical saturation, and consolidation opportunity.

In the prospective period, these signals were discussed at architecture review checkpoints and linked to recorded boundary decisions. These findings support the hypothesis that systematic granularity management is associated with a more structured decision-making process, as reflected in the improved metrics and reduced risk signals during the intervention period. From March 2024 to September 2024, Granulify was used to support proactive granularity assessment:

- **March 2024:** with no new services added (CNS = 51), the system was classified as *No change*, with modest positive indicators (NB = 0.02, TV = 0.16, SS = 0.56), reflecting a stability period;
- **April 2024:** the addition of three microservices (CNS = 54) was classified as *Efficient decomposition* (NB = 0.16, TV = 0.16, SS = 0.99), indicating that the decomposition was associated with positive Net Benefit. In this period, the team focused on internal application improvements, enhancing component reuse and optimizing internal processes, which may help explain the observed positive growth;
- **May 2024:** a warning signal emerged: despite no structural changes (CNS = 54), the system exhibited negative metrics (NB = -0.24, TV = 0.24, SS = 0.01), signaling operational stress even though the period was classified as *No change*;
- **June 2024:** with two additional services (CNS = 56), the system was classified as *Efficient decomposition* (NB = 0.35, TV = 0.35, SS = 0.99), indicating that the actions taken after the May warning were followed by a return to positive indicators;
- **July 2024:** the system transitioned to *Empirical saturation* (NB = 0.01, TV = 0.17, SS = 0.54), reflecting a period in which the benefits of additional decompositions were approaching saturation;

- **August 2024:** with five new services (CNS = 62), the system remained in *Empirical saturation* (NB = -0.05, TV = 0.18, SS = 0.35), indicating that additional fragmentation produced limited Net Benefit relative to the structural variation introduced;
- **September 2024:** a strategic consolidation reduced the number of services by two (CNS = 60) and was classified as *Efficient consolidation*, with strong positive indicators (NB = 0.37, TV = 0.37, SS = 0.99). This move was associated with improvement following the prior saturation pattern.

The intervention evidence was also supported by qualitative artifacts produced during the architecture review process. The May 2024 warning triggered root-cause analysis, and the July–August saturation pattern shifted the decision focus from additional decompositions to boundary consolidation.

The September 2024 consolidation, which reduced the cumulative number of services from 62 to 60 and was classified as *Efficient consolidation*, represents the observable outcome of this decision chain. In operational terms, this consolidation was accompanied by a reduction in KCOST from $56k in August 2024 to $50k in September 2024, an approximate decrease of 10.7%.

According to the architecture review records, this reduction was largely associated with the decommissioning of AWS resources linked to retired workloads, including queues, Lambda functions, inactive service monitoring resources, and database instances retired after consolidation.

The team also reused ECS capacity already running, rather than provisioning additional instances. Therefore, we interpret the KCOST improvement as supporting evidence for the *Efficient consolidation* regime, avoiding the stronger claim that the reduction in service count alone explains the entire cost decrease.

This sequence is consistent with HB_2 because it shows Granulify being used as an intervention support mechanism, rather than merely as a measurement instrument. The transition from *Empirical saturation* in July–August 2024 to *Efficient consolidation* in September 2024 thus provides longitudinal evidence that systematic granularity management can support boundary decisions in an industrial context.

5.5 Threats to Validity

We have presented the results and discussion of the granularity saturation analysis, evaluating the diagnostic and intervention hypotheses. However, it is important to acknowledge the threats to validity that may affect the interpretation of these results.

Identifying and analyzing threats to validity are essential to ensure the reliability of the results and the generalizability of this research. According to the taxonomy proposed by Wohlin et al. (2012), threats are grouped into four categories: *construct validity*, *internal validity*, *conclusion validity*, and *external validity*. Each category is discussed below, together with the mitigation strategies adopted in the context of this study.

5.5.1 Construct Validity

Construct validity concerns the extent to which the operational measures accurately represent the theoretical constructs they intend to measure. In the case of *Granulify*, there is a risk that the metrics and instruments used—such as the Granularity Classification Spectrum and the Granularity Saturation Method—do not fully capture the phenomena of software evolution and modifiability. Additionally, the complexity of the method may lead to low adoption by teams.

To address these concerns, we employed metrics widely recognized in the software engineering literature, sought feedback on the artifacts from experts in the studied organization, and triangulated the quantitative indicators with the qualitative evidence described in Section 4.2.6. This triangulation was used to strengthen the interpretation and traceability of architectural decisions; it was not used to recalibrate the quantitative indicators or to claim independent ground truth for each regime. In future research, automation and the creation of supporting tooling will also be explored to simplify the application of the process and expand its practical applicability.

5.5.2 Internal Validity

Internal validity is related to the causal relationship between the observed variables, considering that the results could have been influenced by external factors, such as organizational changes, deadline pressures, the production transition, or the natural evolution of systems. This threat is particularly relevant because the study follows an action research design in which the first author also served as the lead architect involved in the intervention.

To reduce this risk, contextual conditions were documented throughout the observation period, metric extraction followed a standardized monthly protocol, and interpretation was discussed with technical leads and senior team members, rather than relying on a single observer. The use of architecture review artifacts also provided traceability between the observed signals and subsequent decisions. However, we do not claim that Granulify was the sole cause of the observed improvements; instead, we interpret the results as evidence that Granulify contributed to a more systematic decision-making process.

5.5.3 Conclusion Validity

Conclusion validity refers to the degree to which the conclusions drawn from the data are statistically sound. In this study, there is a risk that the observed relationships between granularity management practices and system quality metrics are coincidental or influenced by confounding variables not considered in the analysis. To reduce overfitting and temporal leakage, threshold calibration was performed using a lagged rolling window, ensuring that each period was classified using only information available before that period.

We also report a window sensitivity analysis to distinguish stable regime signals from classifications that depend on the chosen calibration horizon. Qualitative triangulation strengthens the plausibility and practical interpretation of the findings, but does not eliminate the limitations imposed by

the short time series and the single industrial context. Therefore, the results are interpreted as initial longitudinal evidence, rather than as definitive statistical validation.

5.5.4 External Validity

External validity is associated with the possibility of generalizing the results to other organizational and technical contexts. Since the research was conducted in a single large bank, there is a risk that its findings may not be directly applicable to smaller organizations or other business domains. Based on the operational requirements observed in this study, Granulify should therefore be considered applicable when four minimum conditions are met:

- The system contains at least 10 independently deployable services, or is in a migration stage expected to reach that scale, such that changes in service boundaries produce observable granularity dynamics;
- An observability and engineering data stack is available, covering at least repository, CI/CD, or deployment indicators, code quality, planning/effort, and operational or cost indicators;
- The release or review cadence is compatible with longitudinal snapshots, with monthly snapshots used in this study and equivalent per-release-cycle snapshots possible in other contexts;
- A cadenced architecture review forum exists to interpret the quantitative signal, assess it against practitioner evidence, and record boundary decisions.

Contexts below these conditions can still use the Granularity Classification Spectrum qualitatively, but the GSM regime classification should be treated as exploratory until the measurement and governance base is established. To mitigate this limitation, detailed documentation of the studied context was carried out, facilitating replication by other researchers; future studies in organizations with different characteristics were planned; and open dissemination of the artifacts was chosen, encouraging external evaluations and independent comparisons.

6 Conclusion

This study presented *Granulify*, a comprehensive approach for continuous granularity management in microservice architectures. By integrating theoretical principles with practical processes and artifacts, Granulify enables development teams to systematically monitor, assess, and adjust service boundaries throughout the system lifecycle. The central components of the approach, including the Granularity Classification Spectrum and the Granularity Saturation Method, provide structured artifacts for diagnosing granularity levels and evaluating the effectiveness of decomposition strategies.

6.1 Summary

The research was operationalized through longitudinal action research in a large-scale financial services platform. The study collected and analyzed metrics across the architectural, development, deployment, code quality, and operational dimensions. The results provide initial longitudinal evidence that the absence of continuous granularity management can lead to challenges such as excessive fragmentation, increased maintenance effort, and elevated operational costs.

Furthermore, the empirical study indicates that Granulify can support the identification of granularity-related challenges and the distinction between calibration periods and actionable saturation regimes. By employing quantitative indicators such as Net Benefit, Total Variation, and the Saturation Score, the approach provided insights that informed architectural decisions, supporting teams in iteratively revising service boundaries in response to the system's evolving needs.

6.2 Main Contributions

This research contributes to bridging the gap between the theoretical understanding and practical application of granularity management in microservices. Through longitudinal action research in a financial institution, we showed how Granulify can be operationalized to provide actionable insights. The empirical evidence highlights the dynamic nature of granularity and its multifaceted association with architectural stability, development effort, deployment frequency, code quality, and operational costs. The main contributions include the following:

- The development of the *Granulify* approach, which integrates principles, processes, and artifacts for continuous granularity management.
- The introduction of the **Granularity Classification Spectrum (GCS)**, a structured artifact for diagnosing and categorizing granularity levels.
- The formulation of the **Granularity Saturation Method (GSM)**, which provides quantitative indicators, such as the Saturation Score (SS), for monitoring granularity dynamics and guiding adjustments.
- Initial empirical evidence through longitudinal action research, demonstrating the practical applicability of Granulify in a real-world context.

6.3 Study Limitations

This study presents some limitations. First, it was conducted in a single organizational context, which may restrict the generalization of the results. Furthermore, the approach has not yet been validated across multiple teams or in different domains, a gap that will be addressed in the next phases of the research. For these reasons, the findings should be regarded as indicative evidence; their generalizability depends on replication across multiple domains and organizations. Another limitation concerns the nature of the metrics used, which, although widely recognized in the literature, may not fully capture qualitative aspects, such as the subjective experience of teams or organizational factors that are not directly measurable. Triangulation with governance evidence strengthened the interpretation and traceability of decisions, but did not replace dedicated qualitative studies with broader participant coverage.

6.4 Future Work

In the next steps of this research, we expect this work not only to deepen the understanding of how granularity affects the evolution of microservice-based systems, but also to foster the development of more robust and adaptable methods for architectural management in contemporary software engineering. To this end, future work includes the following:

- Validating the Granulify approach in multiple industrial contexts to assess its generalizability and adaptability.
- Extending Granulify by proposing tools for applying the method in diverse industrial contexts, including different domains and organizational structures, with the possibility of incorporating automation and explainable analysis techniques in future versions.
- Conducting controlled experiments to isolate the effects of specific granularity adjustments on system performance and maintainability.
- Investigating the integration of Granulify with existing DevOps and continuous delivery pipelines to facilitate real-time granularity management.

Additionally, future work will extend the qualitative evaluation of granularity management through dedicated interviews and surveys with software teams, beyond the architecture review discussions used in this study. The goal is to understand how factors such as team satisfaction, ownership clarity, cognitive load, and organizational culture influence and are influenced by granularity decisions. These insights will complement the technical indicators and governance artifacts, offering a more holistic view of continuous granularity management.

Acknowledgements

The authors thank the software engineering teams, architects, and technical leads at the financial institution for their contributions to this study. Their insights during architecture review discussions, sprint retrospectives, and technical checkpoints were essential for interpreting the method's findings and ensuring that the results reflect real-world architectural challenges. We particularly thank the cross-functional teams that provided access to organizational artifacts, metrics, and qualitative observations throughout the 13-month observation period.

This research was conducted with the necessary approvals from the participating organization, including authorization for data collection, analysis, and publication of the results. All identifying information about the institution, specific business capabilities, and individuals was anonymized or generalized to protect confidentiality, while preserving the methodological and empirical validity of the study.

References

Abgaz, Y., McCarren, A., Elger, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yilmaz, M., Buckley, J., and Clarke, P. (2023). Decomposition of monolith applications into microservices architectures: A systematic review. *IEEE Transactions on Software Engineering*, 49(8):4213–4242.

Ait Said, M., Ezzati, A., Mihi, S., and Belouaddane, L. (2024). Microservices adoption: An industrial inquiry into factors influencing decisions and implementation strategies. *International Journal of Computing and Digital Systems*, 15(1):1417–1432.

Avison, D., Lau, F., Myers, M., and Nielsen, P. A. (1999). Action research. *Communications of the ACM*, 42(1):94–97.

Bakhtin, A. (2025). Temporal network analysis of microservice architectural degradation.

Bakhtin, A., Esposito, M., Lenarduzzi, V., and Taibi, D. (2025). Centrality change proneness: an early indicator of microservice architectural degradation.

Becker, A. M. and Lucrédio, D. (2020). The impact of microservices on the evolution of a software product line. In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*, SBCARS '20. ACM.

Behrad, S., Espes, D., Bertin, P., and Phan, C.-T. (2021). Impacts of service decomposition models on security attributes: A case study with 5g network repository function. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE.

Belachew, E. B. (2018). Analysis of software quality using software metrics. *International Journal on Computational Science & Applications*, 8(4/5):11–20.

Bogner, J., Fritzsch, J., Wagner, S., and Zimmermann, A. (2021). Industry practices and challenges for the evolvability assurance of microservices: An interview study and systematic grey literature review. *Empirical Software Engineering*, 26(5).

Bogner, J., Wagner, S., and Zimmermann, A. (2017). Automatically measuring the maintainability of service- and microservice-based systems: a literature review. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, IWSM/Mensura '17. ACM.

Bogner, J., Wagner, S., and Zimmermann, A. (2019). On the impact of service-oriented patterns on software evolvability: a controlled experiment and metric-based analysis. *PeerJ Computer Science*, 5:e213.

Colanzi, T., Amaral, A., Assunção, W., Zavadski, A., Tanno, D., Garcia, A., and Lucena, C. (2021). Are we speaking the industry language? the practice and literature of modernizing legacy systems with microservices. In *15th Brazilian Symposium on Software Components, Architectures, and Reuse*, SBCARS '21. ACM.

Cruz, P., Astudillo, H., Hilliard, R., and Collado, M. (2019). Assessing migration of a 20-year-old system to a microservice platform using atam. In *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, pages 174–181.

Dresch, A., Lacerda, D., and Júnior, J. (2020). *Design Science Research: Método de Pesquisa para Avanço da Ciên-*

cia e Tecnologia. Métodos de Pesquisa. Bookman Editora.

Driessen, F., Ferreira Pires, L., Moreira, J. L. R., Verhoeven, P., and van den Bosch, S. (2024). A quantitative assessment method for microservices granularity to improve maintainability. In Sales, T. P., de Kinderen, S., Proper, H. A., Pufahl, L., Karastoyanova, D., and van Sinderen, M., editors, *Enterprise Design, Operations, and Computing. EDOC 2023 Workshops*, pages 211–226, Cham. Springer Nature Switzerland.

Evans, E. (2003). *Domain-Driven Design Tackling Complexity in the Heart of Software*. Addison Wesley Professional.

Ford, N. (2023). *Building evolutionary architectures*. O'Reilly, Beijing, second edition edition.

Fowler, S. J. (2017). *Production-ready microservices*. O´Reilly, Beijing, first edition edition.

Fritzsch, J., Bogner, J., Wagner, S., and Zimmermann, A. (2019). Microservices migration in industry: Intentions, strategies, and challenges. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE.

Gravanis, D., Kakarontzas, G., Gerogiannis, V., and and (2022). You don't need a microservices architecture (yet): Monoliths may do the trick. In *Proceedings of the 2021 European Symposium on Software Engineering*, ESSE '21, page 39–44, New York, NY, USA. Association for Computing Machinery.

Hasan, M. H., Osman, M. H., Admodisastro, N. I., and Muhammad, M. S. (2023). From monolith to microservice: Measuring architecture maintainability. *International Journal of Advanced Computer Science and Applications*, 14(5).

Hassan, S., Bahsoon, R., and Buyya, R. (2022). Systematic scalability analysis for microservices granularity adaptation design decisions. *Software: Practice and Experience*, 52(6):1378–1401.

Hassan, S., Bahsoon, R., and Kazman, R. (2020). Microservice transition and its granularity problem: A systematic mapping study. *Software: Practice and Experience*, 50(9):1651–1681.

Hassan, S., Bahsoon, R., Minku, L., and Ali, N. (2021). Dynamic evaluation of microservice granularity adaptation. *ACM Transactions on Autonomous and Adaptive Systems*, 16(2):1–35.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York.

Heitlager, I., Kuipers, T., and Visser, J. (2007). A practical model for measuring maintainability. In *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*. IEEE.

Homay, A., Zoitl, A., de Sousa, M., and Wollschlaeger, M. (2019). A survey: Microservices architecture in advanced manufacturing systems. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*. IEEE.

Huang, L., Zhuang, W., Sun, M., and Zhang, H. (2020). Research and application of microservice in power grid dispatching control system. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 1895–1899.

Josélyne, M. I., Tuheirwe-Mukasa, D., Kanagwa, B., and Balikuddembe, J. (2018). Partitioning microservices: a domain engineering approach. In *Proceedings of the 2018 International Conference on Software Engineering in Africa*, ICSE '18. ACM.

Justino, Y., Silva, C., and Duarte, R. (2025). Continuously managing microservice granularity: An evidence-based industrial approach. In *Anais do XXXIX Simpósio Brasileiro de Engenharia de Software*, pages 226–236, Porto Alegre, RS, Brasil. SBC.

Kitchenham, B., Charters, S., et al. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kolny, M. (2023). Scaling up the prime video audio/video monitoring service and reducing costs by 90%.

Mendonca, N. C., Box, C., Manolache, C., and Ryan, L. (2021). The monolith strikes back: Why istio migrated from microservices to a monolithic architecture. *IEEE Software*, 38(5):17–22.

Newman, S. (2021). *Building microservices*. O'Reilly, Beijing, second edition edition.

Nunes, L., Santos, N., and Rito Silva, A. (2019). *From a Monolith to a Microservices Architecture: An Approach Based on Transactional Contexts*, pages 37–52. Springer International Publishing.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058.

Richards, M. (2015). *Microservices vs. service-oriented architecture*. O'Reilly Media.

Salii, S., Ajdari, J., and Zenuni, X. (2023). Migrating to a microservice architecture: benefits and challenges. In *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*. IEEE.

Setiadi, D., Sumitra, T., Karim, A., and Ritzkal (2024). Software quality measurement analysis on academic information systems. *Ingénierie des systèmes d information*, 29(4):1453–1460.

Shaw, M. and Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. An Alan R. Apt book. Prentice Hall.

Taibi, D. and Lenarduzzi, V. (2018). On the definition of microservice bad smells. *IEEE Software*, 35(3):56–62.

Taibi, D., Lenarduzzi, V., and Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Computing*, 4(5):22–32. P1.

Tighilt, R., Abdellatif, M., Trabelsi, I., Madern, L., Moha, N., and Guéhéneuc, Y.-G. (2023). On the maintenance support for microservice-based systems through the specification and the detection of microservice antipatterns. *Journal of Systems and Software*, 204:111755.

Tusjunt, M. and Vatanawood, W. (2018). Refactoring orchestrated web services into microservices using decomposition pattern. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE.

Vera-Rivera, F. H., Gaona, C., Astudillo, H., and and (2021).

Defining and measuring microservice granularity—a literature overview. *PeerJ Computer Science*, 7:e695.

Vernon, V. (2015). *Implementing domain-driven design*. Addison-Wesley, Upper Saddle River, NJ [u.a.], fourth printing edition.

Vural, H. and Koyuncu, M. (2021). Does domain-driven design lead to finding the optimal modularity of a microservice? *IEEE Access*, 9:32721–32733.

Waseem, M., Liang, P., Shahin, M., Di Salle, A., and Márquez, G. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software*, 182:111061.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Computer Science. Springer Berlin Heidelberg.

Zhang, H., Li, S., Jia, Z., Zhong, C., and Zhang, C. (2019). Microservice architecture in reality: An industrial inquiry. In *2019 IEEE International Conference on Software Architecture (ICSA)*, pages 51–60. IEEE.

Zhong, C., Huang, H., Zhang, H., and Li, S. (2022). Impacts, causes, and solutions of architectural smells in microservices: An industrial investigation. *Software: Practice and* ....

Zhou, X., Li, S., Cao, L., Zhang, H., Jia, Z., Zhong, C., Shan, Z., and Babar, M. A. (2023). Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. *Journal of Systems and Software*, 195:111521.

A Appendix: Pseudocode

Algorithm A.1 summarizes the Granularity Saturation Method (GSM) as an implementation-independent procedure. The algorithm receives as input a matrix of raw metrics, metric polarities, a dimension mapping, and a lagged calibration window, and returns the regime assigned to each observed period after the reference baseline.

Signature.

Procedure	$\text{GSM-Regime}(X, G, P, \mathcal{M}, w, n_{\min}, \Theta)$, where $\Theta = (c_{clip}, \tau_{zero}, \tau_{tv}, c_{eff})$.
Input types	$X \in \mathbb{R}^{n \times m}$; $G \in \mathbb{Z}^n$; $P \in \{-1, +1\}^m$; $\mathcal{M} : \mathcal{D} \to 2^{\{1,\ldots,m\}}$; $w, n_{\min} \in \mathbb{Z}_{>0}$; $c_{clip}, \tau_{zero}, \tau_{tv}, c_{eff} \in \mathbb{R}_{>0}$.
Output type	$R \in \text{Regime}^{2..n}$.

Input. $X[i, j]$ is the raw value of metric j in period i; $G[i]$ is the granularity proxy for period i (e.g., CNS); $P[j]$ is the polarity of metric j; $\mathcal{M}$ maps each dimension to the indices of its metrics; w is the maximum number of prior periods used for rolling calibration; and $n_{\min}$ is the minimum number of prior observations required before an actionable classification is issued.

Output. $R[i]$ is the regime assigned to period i, for $2 \le i \le n$.

procedure GSM-Regime$(X, G, P, \mathcal{M}, w, n_{\min}, c_{clip}, \tau_{zero}, \tau_{tv}, c_{eff})$
  for $i \leftarrow 2$ **to** n **do**
    for each dimension $d \in \mathcal{D}$ **do**
      $A \leftarrow \emptyset$
      for each metric $j \in \mathcal{M}(d)$ **do**
        if $|X[i-1, j]| < \tau_{zero}$ **then** $\tilde{r} \leftarrow 0$ ▷ see Eq. (1)
        else $\tilde{r} \leftarrow P[j] \cdot (X[i, j] - X[i-1, j])/|X[i-1, j]|$ ▷ see Eq. (1)
        $r[j, i] \leftarrow \min(c_{clip}, \max(-c_{clip}, \tilde{r}))$ ▷ see Eq. (2)
        $A \leftarrow A \cup \{r[j, i]\}$
      end for
      $r_d[i] \leftarrow \text{Median}(A)$ ▷ see Eq. (3)
    end for
    $NB[i] \leftarrow \text{Median}(\{r_d[i] : d \in \mathcal{D}\})$ ▷ see Eq. (4)
    $TV[i] \leftarrow \text{Median}(\{|r_d[i]| : d \in \mathcal{D}\})$ ▷ see Eq. (5)
    if $TV[i] < \tau_{tv}$ **then** $E[i] \leftarrow 0$ ▷ see Eq. (6)
    else
      $\rho \leftarrow NB[i]/TV[i]$
      if $\rho < -c_{eff}$ **then** $E[i] \leftarrow -c_{eff}$ ▷ extreme NB ratio; see Eq. (6)
      else if $\rho > c_{eff}$ **then** $E[i] \leftarrow c_{eff}$ ▷ extreme NB ratio; see Eq. (6)
      else $E[i] \leftarrow \rho$ ▷ see Eq. (6)
    end if
    $SS[i] \leftarrow (E[i] + 1)/2$ ▷ see Eq. (7)
  end for
  for $i \leftarrow 2$ **to** n **do**
    $W \leftarrow \{k : \max(2, i - w) \le k < i\}$
    if $|W| < n_{\min}$ **then**
      $R[i] \leftarrow$ *Insufficient calibration history*
      continue ▷ insufficient calibration window
    end if
    $\tau \leftarrow Q_{25}(\{TV[k] : k \in W\})$ ▷ see Eq. (8)
    $V \leftarrow \{SS[k] : k \in W \wedge TV[k] \ge \tau\}$ ▷ see Eq. (9)
    $q_{25} \leftarrow Q_{25}(V)$; $q_{75} \leftarrow Q_{75}(V)$ ▷ see Eq. (10)
    if $TV[i] < \tau$ **then** $R[i] \leftarrow$ *Low evidence / stable* ▷ edge case $TV < \tau$
    else if $G[i] = G[i-1]$ **then** $R[i] \leftarrow$ *No change*
    else if $G[i] > G[i-1]$ **and** $SS[i] \ge q_{75}$ **then** $R[i] \leftarrow$ *Efficient decomposition*
    else if $G[i] > G[i-1]$ **and** $SS[i] \le q_{25}$ **then** $R[i] \leftarrow$ *Over-decomposition*
    else if $G[i] < G[i-1]$ **and** $SS[i] \ge q_{75}$ **then** $R[i] \leftarrow$ *Efficient consolidation*
    else if $G[i] < G[i-1]$ **and** $SS[i] \le q_{25}$ **then** $R[i] \leftarrow$ *Harmful consolidation*
    else $R[i] \leftarrow$ *Empirical saturation*
  end for
  return R
end procedure