Accepted for the SoEA4EE'2025 workshop at EDOC 2025. This is the camera-ready author version of the paper, the final version is accessible via Springer.

A Decade of Challenges: A Practitioners Exploratory Study of Microservices Operational Stability

Gabriel Morais[1][0000−0003−1113−7873], Mehdi Adda[1][0000−0002−5327−1758], and Dominik Bork[1,2][0000−0001−8259−2297]

[1] Université du Québec à Rimouski, Lévis, QC, Canada {gabrielglauber.morais, mehdi_adda}@uqar.ca
[2] Business Informatics Group, Technische Universität Wien, Vienna, Austria dominik.bork@tuwien.ac.at

Abstract. Microservices have been around for a decade, but operational challenges remain unsolved. The complexity of operating them has been a persistent issue among practitioners, being a barrier to the proper adoption of the Microservices Architecture (MSA) paradigm. In some extreme cases, this situation led to the failure of MSA adoption and a return to monolithic architectures. Throughout a longitudinal exploratory case study conducted at a large financial organization with over 40 years of experience in developing and operating complex distributed systems, we examine the factors influencing the dependability and sustainability at runtime, i.e., the operational stability of microservices. Our findings highlight the need for systematic approaches to define realistic service-level targets, design cohesive and robust stability strategies, and establish optimal resource utilization, thereby framing a research agenda aligned with industry needs. Ultimately, we ensure the quality of our study by adopting a multivocal approach to data collection, employing systematic data triangulation during analysis, and ensuring transferability by relating our findings to previous studies and collecting feedback from non-participants from a different domain.

Keywords: Microservices Architecture · Operational Stability · Software Dependability · Software Sustainability.

1 Introduction

Over the last decade, microservices architecture (MSA) has driven a paradigm shift in the design and deployment of large, distributed systems, aligning with the continuum of Internet services and inheriting their objectives and challenges [33]. MSA has focused on coping with barriers that prevent fast adaptation to market conditions and evolution [26, 52]. It has fostered independence at all levels as the backbone of adaptable, reusable, scalable and resilient applications, impacting the technology, development and operation processes used when developing software systems [4, 29]. MSA has demonstrated significant benefits in system

design and development, but it remains a complex paradigm when in operation [17, 43, 51]. Due to this complexity, organizations have reverted to monolithic approaches, primarily because of the operational costs and issues with system performance, scalability, and management [44].

The lack of governance and the presence of organizational silos in MSA development, combined with its operational complexity, may cause misalignments that hinder the effective use of this service-oriented paradigm in enterprise architecture (EA) for engineering reliable enterprise systems, leading to weaknesses in EA design, governance, and its ability to support evolving business needs [29, 24]. Besides, migration to microservices may produce hybrid systems composed of microservices and non-microservices components, which are developed and operated using disparate practices [5, 17]. This context leads to specific challenges, such as defining system dependability strategies or determining thresholds for automatic recovery mechanisms that accommodate different architectural paradigms deployed in heterogeneous infrastructures, and using different appraoches [29, 51].

These challenges influence the operational dependability and sustainability of systems, as well as their ability to maintain them at expected levels over time, thereby ensuring their stability. Indeed, systems *operational stability* is achieved through *dependability*, including fault tolerance and avoidance, systems performance and robustness aspects, and *sustainability*, including services' social impact, economic viability, resource consumption optimization, portability, evolvability, and maintainability [1, 23, 22]. Realizing dependability and sustainability requires strategies, measurements, and coordination to be effective [1, 9, 23]. However, these aspects have been overlooked, highlighting the current need for developing novel approaches to handling them within MSA [19, 43, 44].

This research aims to identify the goals, factors, challenges and improvement perspectives related to operational stability of microservices in a microservices migration context, from a practitioner's perspective. We conducted a twelve-month exploratory case study in collaboration with an industrial partner from the finance services domain. Findings reveal that MSA operational stability depends on cross-service coordination and requires holistic approaches to design pragmatic system-wide operational stability strategies, displaying the practical limitations of MSA's core tenets of independence and autonomy. Specifically, practitioners pointed out the challenges of defining realistic end-user service levels, achieving cohesive stability strategies, assessing the robustness of resilience mechanisms, and optimizing resource consumption. These insights contribute to shaping a research agenda grounded in practitioners' latent needs.

Besides, we offer a working definition of operational stability to support understanding stability in this context. Eventually, we evaluated our research against its multivocality, rigour, reflexivity, credibility, and transferability. Artifacts used in and produced by this research are available at this paper's companion repository [28].

The remainder of this paper is organized as follows: Section 2 presents foundational concepts related to operational stability and MSA, as well as a summary

of related works. Section 3 presents the research design. Section 4 reports the findings, and Section 5 discusses their implications and lessons learned. We discuss evaluation in Section 6, and we conclude with final remarks and research perspectives in Section 7.

2 Background and Related Work

This section provides foundational knowledge and related work necessary to understand the research problem and its implications, as well as to contextualize its findings. First, we introduce a working definition of operational stability, followed by the principles and challenges of microservices operations. Finally, we present related work.

2.1 A Working Definition of Operational Stability

Operational stability has been considered a purpose within software stability. It aims to stabilize systems' runtime behaviour, encompassing software evolution, maintenance and runtime aspects [41]. However, a unique definition of operational stability may be inaccurate or incomplete because stability is context-specific and multi-dimensional [40]. A trade-off is to use a working definition built from properties related to the context or case where the stability abilities should be observed. Such a definition provides a conceptual framework to support the understanding of specific stability tenets within the context [41].

In the scope of this paper, we are concerned with operation control processes [49], notably the control of the system's capacity to face failures and service degradation. Focusing on the purpose of stability during the software systems' operation, we constructed a definition for *operational stability* (Definition 1) based on the system's ability to achieve dependability and sustainability.

In this context, the dimensions of dependability and sustainability have specific properties. *Operational dependability* is the ability to avoid system failures that violate established service-level duties. Indeed, dependability is considered the background of building system trustworthiness [11, 45]. *Operational sustainability* is the ability to operate systems cost-effectively, relying on adequate resource consumption. Sustainability encompasses the social, economic, technical, and environmental perspectives of a system, which may support or conflict, requiring their accommodation [12, 23]. Based on this context and dimensions, we define operational stability as follows:

Definition 1. *Operational stability is the ability of a system to maintain its behaviour at a fixed operation level under varying internal and external conditions.*

Here, *system* denotes a software architecture comprising components that interact to deliver a service [1]. *Behaviour* denotes the set of states the system exposes when delivering a service [1]. *Operation level* denotes the degree of compliance with operational specifications, which represent runtime boundaries defined from operational dependability and sustainability attributes for a specific

use environment, establishing the acceptable tolerance threshold for deviation from expected runtime behaviour. The system's *environment* is the set of other systems with which the considered system interacts [1]. The system's structure defines the *varying internal conditions*. The variability in the operational stability of the system environment defines the *varying external conditions*. These properties are often defined throughout contracts that establish the system performance targets as service-level agreements (SLAs) and objectives (SLOs) [10].

2.2 Operating Microservices

Contemporary systems based on cloud microservices adhere to recovery-driven computing (ROC) principles, which focus on building fault-tolerant systems [9, 34]. Microservices are built based on the isolation principle and operated in a scalable infrastructure that enhances redundancy [13, 52]. They are developed in short cycles and rely on approaches that foster automation and tight collaboration between developers and operators, such as DevOps [4, 47]. However, practitioners have identified challenges in handling operational concerns, such as evaluating, establishing baselines and metrics, and effectively monitoring microservices [18, 51].

MSA complexifies system operation due to its distributed nature, which impacts resource consumption, system management, and monitoring [43, 48]. This complexity has been exacerbated by the proliferation of microservices resulting from the MSA migration process [51]. Additionally, organizations transitioning from legacy to MSA systems must manage different paradigms and operational practices in parallel and address the incompatibilities that may arise from operating such hybrid systems. For instance, DevOps has a positive impact on system operational performance and is considered complementary to MSA; however, the adoption of DevOps may not be entirely achieved, and challenges to adopting MSA and DevOps may arise concurrently, influencing each other [3, 47, 51].

The difficulty in comprehensively monitoring microservices leads to the challenge of defining their operational behaviour, which is paramount to supporting automatic resilience mechanisms and assessing performance [48, 51]. It has been achieved by aggregating its historical operation data using application performance monitoring tools and modelling supposed baselines, which are used as comparison points in monitoring activities [18]. Resource usage, load balancing, and availability are the most frequently used monitoring metrics for microservices systems, supporting continuous analysis of application stability and estimation of operational costs [47]. However, frequent microservice releases, fault tolerance, and resilience mechanisms make modelling a microservice's operational normal behaviour difficult [18, 51]. Besides, the monitoring is achieved atomically, i.e., for each microservice individually, which makes establishing the expected behaviour and monitoring microservices compositions complex [18, 29, 47].

2.3 Related Work

In 2017, Soldani et al. [43] reviewed 51 non-academic industrial studies published between 2014 and 2017 to identify the gains and pains of adopting MSA. They unveiled challenges in handling the complexity of distributed systems, data management, and consistency, as well as increased resource consumption. In 2020, Waseem et al. [48] surveyed 106 practitioners using a questionnaire and interviewed six practitioners to identify standard practices and challenges associated with designing, monitoring, and testing microservices. They confirmed the previous challenges and unveiled others related to fault detection and isolation, log exploration, performance assessment and monitoring, and the lack of tools to support operations activities. Between 2021 and 2022, Zhou et al. [51] revisited these challenges to document their evolution. They interviewed 20 practitioners from various industries to analyze their current practices and challenges. Their findings showed that challenges identified by Soldani et al. and subsequent studies remain unresolved. They also elaborated on practical issues in operating microservices, such as excessive technology diversity, difficulties in automating operational processes, and setting thresholds to trigger operational tasks, e.g., recovery mechanisms, calling for formal and methodical approaches.

Previous studies did not focus on runtime dependability and sustainability, and lack a longitudinal, immersive approach, which limits the depth of their observations. Our study complements these prior works by targeting operational stability concerns and considering the extended context of microservices operation within hybrid systems. We also bring depth to the study of these aspects in a real-world scenario through a longitudinal on-site study, enabling a comprehensive understanding and revealing insights that may be previously inaccessible.

3 Case Study Design

We followed the guidelines for the design and execution of observational case studies proposed in [50] and the standard for conducting empirical case studies proposed in [38] to structure this research execution. We engaged with practitioners to incrementally build knowledge about the operational stability challenges they face, using a combination of literature review, interviews, document exploration, and observations, similar to the approach adopted in [31] and [39].

3.1 Objectives and Research Questions

We aimed to qualitatively explore *how* practitioners address MSA operational stability in the context of hybrid systems and *what* challenges they encounter. To meet this goal, we explored the following research questions:

RQ1. Which factors influence the operational stability of hybrid systems?
RQ2. How do these factors influence operational stability?
RQ3. How operational stability practices should be enhanced?

3.2 Defining a Conceptual Framework

We conducted a literature review of taxonomies, ontologies and conceptual frameworks in the fields of MSA and software stability, dependability, and sustainability. Based on them, we built a conceptual framework that bounded our exploration and served as a classification schema to categorize our findings.

3.3 Data Collection

We grounded our study in a critical realism stance, recognizing the role of subjective information from actors while acknowledging the independent structures that influence their actions. We relied on interviews and observations, and approached the research questions from two epistemological perspectives: constructivism and pragmatism [25]. The constructivist perspective ensured the reflection of the role of individual and collective experiences in building an understanding of operational stability, serving as a guideline for interviews. The pragmatic perspective ensured that the role of operational stability practice in extending practitioners' knowledge was considered, guiding our observations.

The first time we met participants, we presented our definition of operational stability (cf. Sec. 2) to avoid any ambiguity or misunderstanding, and asked demographic questions. Table 1 provides a summary of participants' demographics.

Table 1: Participants Demographics

Gender	*Males*											*Females*			
Role	*Business*			*Developer*			*Operator*					*Process Owner*			
YoE-SO	15+		5-10	10-15		15+	10-15			15+		10-15	15+		5-10
YoE-MSA	2-5	5-10	2-5	2-5	5-10	5-10	0-2	2-5	5-10	2-5	5-10	0-2		2-5	5-10
Number	*2*	*1*	*1*	*3*	*1*	*3*	*1*	*2*	*1*	*3*	*4*	*1*	*1*	*1*	*1*
Total Role	**4**			**7**			**11**					**4**			
Total Gender	**23**											**3**			
Total	**26**														

Abbreviations: Years of experience in software operation (YoE-SO), Years of experience in MSA (YoE-MSA).

Interviews – We conducted semi-structured interviews (cf. [28] for interview guides). Each interview involved an interviewee and a researcher who discussed for a duration of at least 30 minutes and a maximum of 60 minutes. Minutes were created and sent to the interviewees for review and validation. Any questions that emerged were recorded in a *questions backlog* to be addressed in subsequent sessions, as in [39]. We conducted four interview cycles from June to November 2024. The number of sessions per cycle varied according to the number of questions in the *questions backlog*.

Observations – We conducted the observations from November 2024 to May 2025 using the protocol proposed in [39], recording the observation period, details, quotes, and follow-up questions. We observed practitioners as they

executed the process of defining system SLAs for eight business functions, each with 4 to 13 services. Stakeholders, operators, and developers varied by function, but the process owner remained the same. Table 2 summarizes our observation execution.

Table 2: Observations Execution

Observed Task	**Participants**	**Duration**	**Times**	**Recurrence**
Process improvement	PO(3)	60 min	3	monthly
SLA definition kick-off	B(1), PO(1), O(1), D(1)	60 min	8	1 / BF
Definition of capability	O(1), D(1)	≈90 min	27	≈3 / BF
SLA negotiation	B(1), PO(1), O(1)	60 min	16	2 / BF
SLA tasks alignment	PO(1), O(1)	60 min	8	1 / BF
Operators-Business alignment	B(4), PO(1), O(11)	60 min	14	2 / month
SLA enactment	PO(1), B(1), O(1)	60 min	8	1 / BF

Abbreviations: Business stakeholders (BS), Process owners (PO), Operators (O), Developers (D), Business functions (BF).

During individual observations, we spent at most 90 minutes with participants, with the duration varying according to the tasks they had to perform. One observer conducted each observation session. We began by asking participants to describe the task they would perform, and then we followed its execution, asking questions throughout the performance. We kept 15 minutes after each session for an open discussion with the participants, during which we collected their thoughts about the tasks they had just performed, their perception of the tasks' utility and effectiveness, as well as the improvements they foresaw. During group meetings, we observed the dynamics of the group as participants aligned their tasks and progressed toward achieving the process. Our goal was to identify how the individual perceptions and improvement ideas observed during individual sessions were conveyed to the group.

3.4 Data Analysis

We conducted a triangulation of the data collected. We compared facts acquired during the interviews against what we observed when practitioners execute operational stability tasks. Then, we examined the organization's documents, including processes documentation, EA-related documents, and architecture decision and operations guides, to corroborate our observations. We sought to confirm perceptions and claims, as well as identify gaps between what practitioners see and do and what is expected. Besides, we reviewed the literature and updated the conceptual framework when we encountered novel insights. We used the framework suggested in [23] to support the data analysis process. This framework identifies factors holistically, relating them to the dimensions under study. It supports the identification of how factors influence each other, representing

dependencies and characterizing them, as dependency can be supportive or conflictual. We relied on UML class diagrams to convey our findings, complementing them with textual descriptions. The final version of the conceptual framework is available at this paper companion [28].

3.5 The Case Studied

To align with our goals, the case study had to encompass complex systems, including microservices that operate concurrently with legacy systems. Additionally, it had to have a significant number of microservices, above 100, and be substantial in the system; thus, we target systems comprising at least 25% of microservices implementing business functionalities.

The selected case is a section of a financial services organization, a concentration of technologies and practices that encapsulate 40 years of system development and operation in a single location. They started MSA-migration in 2018, nowadays, they operate 234 complex business services, comprising 5 to 32 functionalities, totalling 1,300 components, which include legacy services (480 or 69% of business services), microservices (220 or 31% of business services), and infrastructure services (600), such as gateways, databases, and monitoring services. These services are arranged to compose a transactional website serving approximately 1 million users and a transactional intranet portal used by the organization's staff (approximately 20,000 users). We cannot share detailed case information due to the organization's confidentiality policy, but we facilitate some details upon request to assist with reproducing the study.

4 Findings

The findings reveal that operational stability in microservice architectures cannot be achieved in isolation—it demands coordination across system components, directly challenging the MSA principles of independence and autonomy. This tension is evident in the difficulty of defining realistic service-level targets, a problem exacerbated by current methods that rely on localized thresholds rather than system-wide metrics and fail to integrate non-MSA components. These limitations constrain the adoption of advanced dependability techniques, exposing a gap between the capabilities of existing tools and their practical application. Furthermore, the prevalence of longstanding operational challenges in software engineering, as observed in the studied case, such as defining achievable operational targets and developing cohesive strategies, suggests that addressing them requires more than a commitment to MSA tenets and the availability of advanced recovery mechanisms.

Below, we examine these findings through four themes—service level, strategies, mechanisms, and resources—by first identifying the underlying factors (F) shaping stability (RQ1), then analyzing their relationship with stability goals (G) and challenges (C) (RQ2), and finally proposing concrete improvements (P) to address these issues (RQ3).

Table 3: Summary of Key Findings

Observation	***#PA***	***#NPA***
Realistic End-user SLAs	16	2
(G1) End-user satisfaction	23	2
(G2) Optimize operation costs	7	2
(F1) End-user expectations	16	2
(F2) IT capabilities	16	2
(F3) Feasibility of SLA-SLOs	16	2
(F4) Instability level	19	2
(C1) Accommodate end-user expectations and IT capabilities	10	2
(C2) Define operational normal behaviour	10	2
(P1) Systematic approach to establish operation profile	10	2
Cohesive Stragegies	26	2
(G3) Design complementary stability strategies	22	1
(F5) Services' dependencies	22	2
(F6) Services' disruption point	22	2
(F7) Available stability mechanisms	22	2
(F8) Stability mechanisms interplay	22	2
(F9) Compatibility of services and stability mechanisms	13	2
(C3) Assess strategy complementarity	26	2
(C4) Establish operational stability profiles	26	2
(C5) Align stability strategies	26	2
(P2) Methodical approach to align stability strategies	19	2
(P3) Simulation-projection mechanisms	9	1
Robustness of Resilience Mechanisms	17	2
(G4) Choose the right mechanism for the right scenario	17	2
(F10) Mechanism configuration	15	2
(F11) Mechanism reliability	17	2
(C5) Benchmark robustness	15	2
(C6) Evaluate mechanisms in real-life conditions	15	2
(C7) Assess the mechanism's robustness when facing cascading failures	15	1
(P4) Methods to identify mechanisms disruption points	17	2
Optimal Resource Consumption	22	2
(G5) Saving on stability mechanisms costs	8	2
(F12) Billing model	4	1
(F13) Consumption goals-limits	22	2
(F14) Outages-failure resource costs	8	1
(C8) Accommodate SLAs-SLOs and resource consumption	22	2

Abbreviations: Operational stability goal (G), factors (F), challenges (C), propositions (P), participants (PA), non-participant (NPA).

4.1 Realistic end-user service level agreements

Participants (16) highlighted the theme of service level as a central component in understanding and achieving operational stability. Notably, the importance of

defining realistic service level targets that align with end-user expectations (F1) and IT capabilities (F2), as well as setting feasible targets (F3), is emphasized. All operators and business participants (19) indicated service instability level (F4) as the main factor considered when defining a service level target. In this context, the instability level acts as a risk indicator. By doing so, the objective is to achieve high end-user satisfaction (G1) while optimizing costs (G2).

The factors associated with the need for realistic end-user service level agreements align with what the literature identified as the need to unveil and understand the goals and requirements related to operational stability concerns [41]. It involves identifying stakeholders and their goals to derive views, requirements, and stability strategies, aiming to reach a consensus by accommodating stakeholders' needs, objectives and IT capabilities. Once a common view is reached, implementation strategies can be derived [11, 20]. These end-user SLAs are then transformed into local SLOs, which must be met to comply with the agreed SLAs [16, 46]. This factor aligns with the social perspective on sustainable systems [23], as the systems provide trustworthy services to end-users.

Participants (10) stressed the challenges of balancing end-user expectations and IT capabilities (C1), which necessitate baselining system capabilities (C2), aligning with longstanding issues in MSA [18, 43, 51]. A proposition is to develop systematic approaches to establish local and system operation profiles, which requires benchmarking SLAs-SLOs by defining rules and baselines, allowing them to measure their achievement system-wide, similar to requests made in [51].

4.2 Cohesive Strategies

All process owners, operators, and developers emphasized the importance of unveiling service dependencies at runtime (F5), arguing that services, both legacy and microservices, are often developed in silos with limited interaction between teams, resulting in recurring mismatches in architectural documents between development and operations, as noted in [30, 29]. Similarly, they pointed out the services' stability profile (F6), the available stability mechanisms (F7), and their interplay (F8) as unavoidable factors in designing complementary stability strategies (G6). They consider aligned objectives as essential because the predominance of disparate stability objectives within services may prevent the achievement of dependability and sustainability at the system level. Furthermore, they noted that the service nature, i.e., legacy or microservice, introduces compatibility concerns (F9). Operators (7) and developers (6) indicated that assessing stability strategies is a challenge (C3) that requires reliable system runtime models, identifying the trigger patterns of stability mechanisms, and a dedicated test environment, aligning with known limitations in testing microservices and runtime architecture [18, 29, 35, 43].

All participants emphasized understanding the system stability profile (C4), which identifies service properties that affect operational stability. While C4 is related to C2 in terms of modelling service runtime behaviour, C2 focuses on the system's standard operational capabilities without considering SLA and SLO targets, while C4 examines behaviour changes leading to SLA and SLO

violations. In all, they are concerned with the negative impact of contradictory stability actions (C5).

The case primarily relies on a 'fix-as-fails' strategy, addressing faults as they occur. It manages operational stability within the ITIL framework for service operations, particularly through incident management and problem management practices [2]. Automatic and manual monitoring are key elements of its operational strategy, which focus on the velocity of detecting, alerting, classifying, and fixing outages. Two operators mentioned the low level of "intelligent monitoring" and advocated for the adoption of observability practices [27]. Besides, six operators pointed out the annoyance caused by improper alert thresholds, relating this issue to the lack of practical guidance and operational knowledge required to set the fit values when configuring them. Preventive strategies using automatic resilience processes were observed; however, four operators noted that they have, occasionally, been a source of instability, as reported in [51].

Generally, participants (19) emphasized the lack of systematic and objective methods, such as data-based or test-based approaches, indicating that current practices foster ad-hoc and approximate solutions, which we observed when they executed operational stability-related processes. To address these challenges, they proposed benchmarking strategies for alignment and complementarity. Thus, the proposition of developing a methodical approach to align stability strategies (P2) emerged. Some of the participants (9) envisioned a simulation system (P3) that could project operational behaviour without deploying the system, resilience and recovery mechanisms in a physical structure, which would allow them to conduct tests and experiment with stability strategy variants. P2 closely align with recent research on applying Digital Twins techniques to MSA and cloud applications to avoid infrastructure costs during test activities [7, 36].

4.3 Robustness of Resilience Mechanisms

The robustness of resilience mechanisms encompasses their adaptability, strength, and capacity to keep the system in the desired state when activated. Robust resilience mechanisms within cohesive stability strategies relate to technical sustainability [23]. It directs the design of adaptable and resilient systems, influencing their long-term use [12]. Participants (15) were concerned with the adequacy of resilience mechanism configurations (F10), mainly in how minimal and maximal values are established and in identifying variations between expected and actual mechanisms' behaviour, echoing the challenge of defining adequate triggers identified in [51]. Besides, participants (17) identified the reliability of resilience mechanisms (F11) as a critical decision factor. Indeed, considering that stability mechanisms can face disruption helps practitioners analyze and make informed decisions when designing stability strategies and defining realistic SLAs.

These considerations remained theoretical in the case without practical application. Developers (7) and operators (8) acknowledged their importance but indicated that they did not apply any specific method to assess them. They argued that they face a lack of standards to benchmark robustness (C5) and the

need for a test environment representative of real-life conditions (C6) to conduct mechanism evaluation in a trial-and-error approach, which is the ad-hoc technique we observed some of them used (2 operators). They also indicated the challenge of assessing mechanisms' robustness when facing cascading failures (C7), from the viewpoint of mechanisms' interaction. This perspective differs from the one identified in [43] and [51], where participants were concerned with the cascading effect of service failures.

Therefore, participants (17) identified the need to pinpoint disruption points in mechanisms (P4), i.e., when mechanisms are overwhelmed, and to track variations over time, allowing them to evaluate the mechanism's reliability and outage risks. Similar to experimenting with stability strategies, they suggested using simulators (P3) in place of real test environments to enhance mechanisms manipulation and experiments.

4.4 Optimal Resource Consumption

Business participants pointed out the cost of stability mechanisms as a critical factor in ensuring the economic sustainability of stability strategies. This cost is directly related to resource consumption and the platform provider's billing model (F12). Operators and developers noted the case of using autoscaling resilience mechanisms, where the costs vary depending on the type of service being scaled. Also, resource consumption is budgeted yearly, imposing a consumption limit goal (F13). However, operators and developers noted the lack of knowledge of real costs as they are managed outside their teams.

Nevertheless, all operators noted that they have extensive knowledge of normal resource consumption levels, which they use to optimize operations. We observed the practical use of this knowledge when operators manually adjust resources in production to adapt to usage peaks. Similarly to the definition of stability strategies, all operators and developers highlighted the influence of connections between microservices and legacy services on resource consumption induced by inadequate choices (F9 and C5). For instance, we observed that operators closely follow the scalability of microservices that depend on legacy parts that are not scalable, both for technical and business reasons. All developers and operators noted the impracticability of generalizing autonomous resource management through the auto scale-up and down practice in this context.

Building dependable systems is considered a priority by business and process owners to achieve end-user satisfaction (G1). However, it is likely to conflict with resource savings and consumption limits aimed by the same business participants (G5). Indeed, the activation of stability mechanisms helps ensure dependability, but it also increases operational costs (F14). Operators and developers nuanced this conflict; for them, these costs depend on the number and activation rate of stability mechanisms, which in turn depend on the particularities of the service and its operational stability profile (F6).

Operators, developers, and process owners highlighted the alignment between SLAs, strategies, and implemented mechanisms and their consumption levels as

essential conditions for achieving operational stability but acknowledged the difficulty in concretizing this alignment (C8). They emphasized that these conditions are addressed individually within each service. It is worth noting that in this case study, resource consumption limits are allocated to each service individually, but SLAs may overlap between services. The lack of wide alignment at the system level is considered a source of fragility for operators and an economic risk for process owner participants. One operator summarized the situation: "*The biggest challenge I face is trusting resilience mechanisms. I lack a complete understanding of the interplay between all the system's parts, I cannot ensure these mechanisms will align with what is required when handling a service disruption, in both their capability to keep the service up and to fit in the expected costs.*"

These factors echo the challenges of managing microservice resource consumption unveiled by previous research [43, 48, 51]. Yet, here, we observed this concern in the context of supporting processes, not within microservices. Again, the proposal of a simulation and projection mechanism (P3) to support the design and assessment of resource consumption was mentioned, which reinforces previous findings on the practitioners' need for systematic and virtualized approaches to design effective MSA operational strategies [19, 35].

5 Discussion

The findings of this study suggest a continuity in the operational challenges experienced by practitioners, aligning with existing literature in software systems operations and MSA challenges. The need for reachable targets, systematic approaches and assessment methods as fundamental elements for building customers' trustworthiness has been suggested as a paramount factor in designing, implementing and operating reliable systems since the 1990s [11]. Similarly, economic sustainability is a constant in product development and operation [42]. The novelty lies in the perspective of resource consumption optimization due to environmental concerns. However, even here, the results have a direct impact on the economics of the product, thus on business motivations.

Also, we unveiled novel insights into MSA and hybrid-system interplay, notably the significant role of cohesion between these system parts, which may, in some cases, be unattainable due to their intrinsic characteristics. This insight allows us to advance the theoretical understanding of microservices operations within hybrid systems by linking this phenomenon to two theorized phenomena in distributed software engineering: the consensus problem [6, 15] and the mixed-criticality problem [8]. On the one hand, system parts are independent components that collaborate to achieve a goal, requiring consensus when deciding on an event. On the other hand, each of these parts may be characterized by a different business or technical criticality. Beyond that, we identified collaboration and coordination factors behind those pointed out by the participants. The central place of coordination and the need for enforced alignment between the system's parts may encounter the organizational silos observed in MSA practices [29].

Therefore, questions arise about the practicability of the MSA's share-nothing philosophy.

Besides, we found that the concept of resilience in MSA extends beyond the technical implementation of mechanisms such as autoscaling or redundancy [18]. Here, the critical issue is not the availability of advanced resilience mechanisms but the lack of a unified approach to define, quantify, and enforce reliability objectives across a multi-service and multi-architecture system. Therefore, the challenge lies not in deploying resilience mechanisms, but in designing them to operate as a cohesive whole—a problem that existing practices and tools alone do not fully address. The shift from technical capability to systemic alignment represents the unaddressed frontier in MSA operational stability, where the complexity stems less from tooling and more from architectural and organizational misalignment. This frontier affects the EA's ability to effectively build service-oriented systems that align with the enterprise's goals.

We interpreted our study findings from the perspective of engineering practices. Therefore, there is room for alternative explanations. One plausible alternative is the effect of using a specific instance or combination of methodologies and frameworks, e.g., Agile, DevOps, and Waterfall, as the common factor identified in the above challenges is the coordination of system parts that directly depends on teams' coordination, which is influenced by the development and operation methodology adopted [37]. Indeed, the operational challenges observed may originate from insufficient DevOps adoption or its fragmented implementation. While DevOps, with its emphasis on cross-functional collaboration, is often viewed as synergistic with MSA, it is not inherently required for microservice development and operation [51, 47]. The case studied exemplifies this: the organization's partial adoption of DevOps with a segregation of development and operations teams may amplify coordination challenges. In such contexts, the overlapping struggles of MSA and DevOps adoption may create a feedback loop, where gaps in one domain exacerbate difficulties in the other [44, 47]. Nevertheless, a system-wide practices alignment seems required, which necessitates adapting development methods to encompass the entire system, including non-microservices components.

Another perspective for explanations may be the lack of literacy among practitioners on specific aspects of MSA [32]. Likewise, the impact of transferring practices from previous paradigms to MSA may influence some of the identified issues, as this practice may generate MSA anti-patterns [14].

A key lesson in conducting this study is flexibility, as the unpredictable nature of software failures, the substantial workload of operators, and the tight schedules of business stakeholders often disrupted this research execution, leading to context-shifting that needed restarting interviews and observations, which demanded patience and rigour. A second lesson is transparency. Industrial studies may face issues of secrecy that restrict researchers' access to precise descriptions of processes, goals, or motivations [21]. Software operation is sensitive to them, as they fear reputational damage by exposing weaknesses. Despite having worked with this organization on other research projects, building confidence and trust

with participants remained a pivotal and time-consuming process. At the first contact with participants, we conveyed that our purpose was not to criticize individual practices or spy but to understand their reality and unveil the causes of the challenges they face, which they could assess by reviewing anonymized interview minutes, observation protocols, and this paper. Some initially reticent participants became more open to the study's conclusions, thanks to the policy of transparency we adopted. A third lesson is managing participants' subjective perceptions. We must distinguish between criticisms driven by discontent with the organization and objective assessments of operational reality. Data triangulation and immersive, on-site observations have proven essential in navigating this issue.

6 Evaluation

We relied on the quality criteria for evaluating exploratory case studies proposed in [38]. Therefore, we evaluated our study against its multivocality, rigour, reflexivity, credibility, and transferability.

We achieved *multivocality* by including participants with both technical and non-technical profiles, different backgrounds, and diverse viewpoints on operational stability-related processes. We maintained *rigour* by employing a systematic data collection and analysis approach, including a data triangulation strategy. Additionally, we maintained a "questions backlog," which gauged the need for more data and analysis; thus, being our saturation indicator. We achieved *reflexivity* by continuously auto-analyzing our interactions with participants, particularly considering the risks of unintentionally influencing their responses or actions, as we brought external knowledge and experience that may not have been present in the case. When in doubt, we cross-checked interview minutes and observation journals with those of other participants. If uncertainty persisted, we re-interviewed the participant to check for any influence; as a final recourse, we excluded the data from the analysis. We maintained the *credibility* of our conclusions through the multivocal and triangulated data collection and analysis approach we employed. Finally, *transferability* refers to the ability of a study's results to plausibly apply to other contexts. Indeed, case studies aim to produce theoretical generalizations, creating transferable concepts applicable to various contexts. To assess transferability, we first related our findings to existing literature. We then shared our results with non-participant practitioners from the health insurance (1) and real estate (1) domains to check whether our insights were recognizable and familiar to them. Their feedback confirmed that our findings resonated, unveiling a set of common concerns in the industry.

7 Conclusion and Future Work

This study examined challenges to the operational stability of microservices within hybrid systems through a longitudinal exploratory case study. We found that the coherence and coordination of stability attributes among a system's

components are the primary factors influencing the effectiveness of operational stability strategies. Additionally, operational stability attributes are complex to estimate, underscoring the need for systematic approaches. Without such systematic approaches to translate high-level operational stability goals and requirements into cohesive, cross-service strategies, even well-implemented stability mechanisms risk failing to deliver effective system dependability and sustainability. This gap between localized stability tactics and holistic stability governance is where the next frontier of microservice operation lies. Future work will focus on developing metrics, indicators and interpretation models to support operational stability decisions in MSA.

References

1. Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE transactions on dependable and secure computing **1**(1), 11–33 (2004)
2. AXELOS: Itil foundation, itil 4 edition (2019)
3. Azad, N., Hyrynsalmi, S.: Devops critical success factors—a systematic literature review. Information and Software Technology **157**, 107150 (2023)
4. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Migrating to cloud-native architectures using microservices: an experience report. In: Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers 4. pp. 201–215. Springer (2016)
5. Balalaie, A., Heydarnoori, A., Jamshidi, P., Tamburri, D.A., Lynn, T.: Microservices migration patterns. Software: Practice and Experience **48**(11), 2019–2042 (2018)
6. Barborak, M., Dahbura, A., Malek, M.: The consensus problem in fault-tolerant computing. ACM Comput. Surv. **25**(2), 171–220 (Jun 1993). https://doi.org/10.1145/152610.152612
7. Borsatti, D., Cerroni, W., Foschini, L., Grabarnik, G.Y., Poltronieri, F., Scotece, D., Shwartz, L., Stefanelli, C., Tortonesi, M., Zaccarini, M.: Modeling digital twins of kubernetes-based applications. In: 2023 IEEE Symposium on Computers and Communications (ISCC). pp. 219–224 (2023). https://doi.org/10.1109/ISCC58397.2023.10217853
8. Burns, A., Davis, R.I.: A survey of research into mixed criticality systems. ACM Computing Surveys (CSUR) **50**(6), 1–37 (2017)
9. Candea, G., Brown, A.B., Fox, A., Patterson, D.: Recovery-oriented computing: Building multitier dependability. Computer **37**(11), 60–67 (2004)
10. Comuzzi, M., Kotsokalis, C., Spanoudakis, G., Yahyapour, R.: Establishing and monitoring slas in complex service based systems. In: 2009 IEEE International Conference on Web Services. pp. 783–790. IEEE (2009)
11. Dale, C.: Software reliability assessment–the need for process visibility. IFAC Proceedings Volumes **23**(6), 77–82 (1990)
12. David, I., Bork, D., Kappel, G.: Circular systems engineering. Software and Systems Modeling pp. 1–15 (2024)
13. Dragoni, N., Lanese, I., Larsen, S.T., Mazzara, M., Mustafin, R., Safina, L.: Microservices: How to make your application scale. In: Perspectives of System Informatics: 11th International Andrei P. Ershov Informatics Conference, PSI

2017, Moscow, Russia, June 27-29, 2017, Revised Selected Papers 11. pp. 95–104. Springer (2018)
14. Farsi, H., Allaki, D., En-Nouaary, A., Dahchour, M.: Dealing with anti-patterns when migrating from monoliths to microservices: Challenges and research directions. In: 2023 IEEE 6th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech). pp. 1–8. IEEE (2023)
15. Fischer, M.J.: The consensus problem in unreliable distributed systems (a brief survey). In: International conference on fundamentals of computation theory. pp. 127–140. Springer (1983)
16. Frey, S., Reich, C., Lüthje, C.: Key performance indicators for cloud computing slas. In: The fifth international conference on emerging network intelligence, EMERGING. pp. 60–64 (2013)
17. Fritzsch, J., Bogner, J., Wagner, S., Zimmermann, A.: Microservices migration in industry: intentions, strategies, and challenges. In: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 481–490. IEEE (2019)
18. Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L.E., Pahl, C., Schulte, S., Wettinger, J.: Performance engineering for microservices: Research challenges and directions. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. p. 223–226. ICPE '17 Companion, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3053600.3053653
19. Jagadeesan, L.J., Mendiratta, V.B.: When failure is (not) an option: Reliability models for microservices architectures. In: 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 19–24 (2020). https://doi.org/10.1109/ISSREW51248.2020.00031
20. Kieninger, A., Westernhagen, J., Satzger, G.: The economics of service level engineering. In: 2011 44th Hawaii International Conference on System Sciences. pp. 1–10. IEEE (2011)
21. Kontio, J., Bragge, J., Lehtola, L.: The focus group method as an empirical tool in software engineering. In: Guide to advanced empirical software engineering, pp. 93–116. Springer (2008)
22. Koziolek, H.: Sustainability evaluation of software architectures: a systematic review. In: Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS. pp. 3–12 (2011)
23. Lago, P., Koçak, S.A., Crnkovic, I., Penzenstadler, B.: Framing sustainability as a property of software quality. Communications of the ACM **58**(10), 70–78 (2015)
24. Lankhorst, M.: Enterprise architecture at work: modelling, communication, and analysis. Springer (2017)
25. Lawani, A.: Critical realism: what you should know and how to apply it. Qualitative research journal **21**(3), 320–333 (2021)
26. Lewis, J., Fowler, M.: Microservices (2014), `https://martinfowler.com/articles/microservices.html`
27. Li, B., Peng, X., Xiang, Q., Wang, H., Xie, T., Sun, J., Liu, X.: Enjoy your observability: an industrial survey of microservice tracing and analysis. Empirical Software Engineering **27**, 1–28 (2022)
28. Morais, G., Adda, M., Bork, D.: Companion repository, `https://bit.ly/edoc87`, companion repository for the paper titled "A Decade of Challenges: A Practitioners Exploratory Study of Microservices Operational Stability"

29. Morais, G., Adda, M., Bork, D.: Breaking down barriers: Building sustainable microservices architectures with model-driven engineering. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems. pp. 528–532 (2024)
30. Morais, G., Lemelin, E., Adda, M., Bork, D.: Enhancing api labelling with bert and gpt: An exploratory study. In: International Conference on Enterprise Design, Operations, and Computing. pp. 169–182. Springer (2024)
31. Morais, G., Lemelin, E., Adda, M., Bork, D.: Large language models for api classification: An explorative study. In: 2025, EASE (2025)
32. Nogueira, V.L., Felizardo, F.S., Amaral, A.M., Assunção, W.K., Colanzi, T.E.: Insights on microservice architecture through the eyes of industry practitioners. In: 2024 IEEE International Conference on Software Maintenance and Evolution (ICSME). pp. 765–777. IEEE (2024)
33. Oppenheimer, D., Patterson, D.A.: Architecture and dependability of large-scale internet services. IEEE Internet Computing **6**(5), 41–49 (2002)
34. Patterson, D., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., et al.: Recovery-oriented computing (roc): Motivation, definition, techniques, and case studies. Tech. rep., Citeseer (2002)
35. Pietrantuono, R., Russo, S., Guerriero, A.: Testing microservice architectures for operational reliability. Software Testing, Verification and Reliability **30**(2), e1725 (2020)
36. Raghunandan, A., Kalasapura, D., Caesar, M.: Digital twinning for microservice architectures. In: ICC 2023 - IEEE International Conference on Communications. pp. 3018–3023 (2023). https://doi.org/10.1109/ICC45041.2023.10279802
37. Raj, P., Sinha, P.: Project management in era of agile and devops methodlogies. In: International Conference on Applied Sciences. vol. 9, pp. 1024–1033 (2015)
38. Ralph, P., bin Ali, N., Baltes, S., Bianculli, D., Diaz, J., Dittrich, Y., Ernst, N., Felderer, M., Feldt, R., Filieri, A., de França, B.B.N., Furia, C.A., Gay, G., Gold, N., Graziotin, D., He, P., Hoda, R., Juristo, N., Kitchenham, B., Lenarduzzi, V., Martínez, J., Melegati, J., Mendez, D., Menzies, T., Molleri, J., Pfahl, D., Robbes, R., Russo, D., Saarimäki, N., Sarro, F., Taibi, D., Siegmund, J., Spinellis, D., Staron, M., Stol, K., Storey, M.A., Taibi, D., Tamburri, D., Torchiano, M., Treude, C., Turhan, B., Wang, X., Vegas, S.: Empirical standards for software engineering research (2020), `https://arxiv.org/abs/2010.03525`
39. Roehm, T., Tiarks, R., Koschke, R., Maalej, W.: How do professional developers comprehend software? In: 2012 34th International Conference on Software Engineering (ICSE). pp. 255–265. IEEE (2012)
40. Salama, M., Bahsoon, R.: A taxonomy for architectural stability. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. pp. 1354–1357 (2016)
41. Salama, M., Bahsoon, R., Lago, P.: Stability in software engineering: Survey of the state-of-the-art and research directions. IEEE Transactions on Software Engineering **47**(7), 1468–1510 (2019)
42. Sauvé, J., Marques, F., Moura, A., Sampaio, M., Jornada, J., Radziuk, E.: Sla design from a business perspective. In: Ambient Networks: 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2005, Barcelona, Spain, October 24-26, 2005. Proceedings 16. pp. 72–83. Springer (2005)
43. Soldani, J., Tamburri, D.A., Van Den Heuvel, W.J.: The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software **146**, 215–232 (2018)

44. Su, R., Li, X., Taibi, D.: From microservice to monolith: A multivocal literature review. Electronics **13**(8), 1452 (2024)
45. Tsiakis, T., Katsaros, P.: Hands on dependability economics. In: 2009 Second International Conference on Dependability. pp. 117–121 (2009). https://doi.org/10.1109/DEPEND.2009.24
46. Walter, J., Okanović, D., Kounev, S.: Mapping of service level objectives to performance queries. In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion. pp. 197–202 (2017)
47. Waseem, M., Liang, P., Shahin, M.: A systematic mapping study on microservices architecture in devops. Journal of Systems and Software **170**, 110798 (2020)
48. Waseem, M., Liang, P., Shahin, M., Di Salle, A., Márquez, G.: Design, monitoring, and testing of microservices systems: The practitioners' perspective. Journal of Systems and Software **182**, 111061 (2021)
49. Washizaki, H. (ed.): Guide to the Software Engineering Body of Knowledge (SWEBOK). IEEE Computer Society, version 4.0 edn. (2024), available online: `https://www.computer.org/swebok`
50. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Computer science, Springer Berlin Heidelberg (2014)
51. Zhou, X., Li, S., Cao, L., Zhang, H., Jia, Z., Zhong, C., Shan, Z., Babar, M.A.: Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry. Journal of Systems and Software **195**, 111521 (2023)
52. Zimmermann, O.: Microservices tenets: Agile approach to service development and deployment. Computer Science-Research and Development **32**, 301–310 (2017)