electronics

MDPI

Review

From Microservice to Monolith: A Multivocal Literature Review †

Ruoyu Su *, Xiaozhou Li and Davide Taibi

M3S, Faculty of Information Technology and Electrical Engineering, University of Oulu, Pentti Kaiteran Katu 1, 90570 Oulu, Finland; xiaozhou.li@oulu.fi (X.L.); davide.taibi@oulu.fi (D.T.)

* Correspondence: ruoyu.su@oulu.fi

† This paper is an extended version of our paper published in Su, R.; Li, X.; Taibi, D. Back to the Future: From Microservice to Monolith. In Proceedings of the 5th International Conference on Microservices (Microservices 2023), Pisa, Italy, 10–12 October 2023.

Abstract: Recently, the phenomenon of switching back from microservice to monolith has increased in frequency, leading to intense debate in the industry. In this paper, we conduct a multivocal literature review to investigate reasoning and key aspects to pay attention to when switching back and analyze other practitioners' opinions. The results show four cases of switching back from microservice to monolith: Istio control plane, Amazon Prime Video monitoring service, Segment, and InVision. The five main reasons that led to switching back are cost, complexity, scalability, performance, and organization. During the switching back process, six key aspects need to be addressed: (1) stopping the development of more services, (2) consolidating and testing paths, (3) unifying data storage, (4) implementing the message bus principle, (5) giving up diverse techniques, and (6) learning to use modular design principles. As to the practitioners' opinions, they had mixed views about the switching back phenomenon. However, most thought that switching back required consideration of the actual system situation and principles. These results pave the way for further research and guide researchers and companies through the process of switching back from microservice to monolith.

Keywords: microservice; monolith; switching back; multivocal literature review; practitioner

Citation: Su, R.; Li, X.; Taibi, D. From Microservice to Monolith: A Multivocal Literature Review. *Electronics* **2024**, *13*, 1452. https://doi.org/10.3390/electronics13081452

Academic Editor: Manuel Mazzara

Received: 14 March 2024
Revised: 9 April 2024
Accepted: 10 April 2024
Published: 11 April 2024

1. Introduction

Microservices have become an essential architectural style due to their independent and decentralized nature [1,2]. In recent years, microservices have gained significant traction, particularly within industry [3]. Large companies like Netflix, Amazon, and Spotify have also adopted microservice architectures, and more enterprises are migrating their systems to microservices [4,5]. They expected to take advantage of the benefits of microservices, such as independent development, deployment, and scalability, in order to help their systems solve current problems, improve the systems' quality, or enable easier software maintenance [6,7]. However, many companies have not achieved the expected benefits of migrating to microservices, although several have significantly improved their performance and independence [8]. Research and practice in recent years have also proved Fowler and Lewis's [9] point that microservice architecture is not a silver bullet and that it faces several challenges at different stages of the process [10,11].

Recently, there has been a phenomenon in the industry of systems switching back from microservice to monolith. One example is Amazon Prime Video monitoring service, one of the world's most extensive streaming services, serving millions of customers worldwide. The report claimed that switching from a distributed microservice architecture to a monolithic application helps achieve grander scale, resilience, and lower costs, solving the questions of higher cost and scaling bottlenecks [12]. This report generated much discussion among practitioners. Although microservices are becoming more popular, even large companies like Amazon have made rollbacks from microservices because of some shortcomings of microservices. Moreover, there are also discussions of the advantages and

disadvantages of microservices and monolithic architecture. Monolithic architecture is easy to develop and simple to deploy, but it is difficult to scale and has complex maintenance and low reliability and availability [13]. In contrast, microservices architecture is easy to scale and maintain and has high reliability and availability, but it requires more complex deployment and autonomy [13].

To research this phenomenon of switching back from microservice to monolith, we aim to investigate the status of switching back examining the reasons, methods, techniques, and opinions involved. For this purpose, we conduct a multivocal literature review. Specifically, we investigate (1) reasons for companies switching back from microservice to monolith, (2) key aspects to be aware of during the switching back process, and (3) opinions of the other practitioners about this switching back phenomenon. We identified one paper out of 295 in white literature and 31 articles out of 198 in grey literature; in the end, 32 articles were selected for study.

The results show four cases of switching back from microservice to monolith: Istio control plane, Amazon Prime Video monitoring service, Segment, and InVision. The five main reasons that led to the switching back are cost, complexity, scalability, performance, and organization. During the switching back process, six key aspects need to be aware: (1) stop developing more services, (2) consolidate and test paths, (3) unify data storage, (4) implement message bus principle, (5) give up diverse techniques and (6) learn to use modular design principles. From the practitioners' opinions, they had mixed views about the switching back phenomenon. However, most thought the switching back needed considering the actual system situation and principles. Results provide researchers with a comprehensive view of the actions and perspectives of organizations and practitioners in switching back from microservice to monolith.

This paper is structured as follows: Section 2 outlines the research methodology adopted in the study. In Sections 3 and 4, we summarize and discuss the obtained results. Section 5 identifies the threats to validity. Finally, in Section 6, we present an overview of related work, and Section 7 provides the conclusion.

2. Methodology

In this work, we aim to understand the state of the art regarding the reasons and methods facilitating switching back from microservice to monolith and practitioners' opinions and advice towards such practices. To such an end, we conducted a multivocal literature review (MLR) based on the guidelines defined by [14]. An MLR is a combination of two parts: (1) a systematic literature review (SLR) on the academic literature (white) published in journals or conferences, and (2) one on the grey literature, e.g., blog posts, social media posts, and videos [14]. MLRs are popular and can provide an extra layer of insight, especially toward understanding newly emerging trends in the software industry.

In this section, we define the objectives and research questions of the study (Section 2.1) and report our search strategy (Section 2.2). Moreover, we outline the data extraction and analysis (Section 2.3) and share our work process replication package for this study (Section 2.4).

2.1. Goals and Research Questions

The goals of this paper are three-fold. First, we investigate the main reasons that encourage notions of switching back from microservice to monolith, especially when the popularity of microservice is still at its pinnacle. It is critical to find any specific industrial cases that support such phenomena with explicit reasons provided. Second, we also investigate which critical activities or aspects of the reverse switching back process deserve special attention. Furthermore, we are interested in other opinions from practitioners regarding such phenomena.

Based on the goals, we defined the following research questions ($\mathbf{RQ}_s$):

- $\mathbf{RQ}_1$. What are the reasons for switching back from microservice to monolith?
- $\mathbf{RQ}_2$. What are the key aspects to pay attention to during the switching back process?

- **RQ**$_3$. What are the opinions of the other practitioners regarding such switching back?

2.2. Search Strategy

The search strategy of an MLR consists of two parts, namely the SLRs conducted on both the white and the grey literature. This includes the definition of search strings, the choice of bibliographic sources, the determination of the inclusion and exclusion criteria, and the search and selection process. The search strategy is depicted in Figure 1.

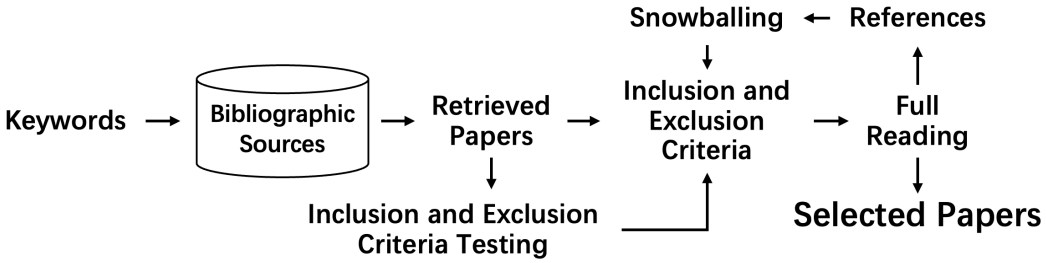

Figure 1. The search and selection process.

Search String. We identified the main terms to be considered based on our research goals and questions. As such, we started with the terms "microservice" and "monolith" to investigate the phenomena of switching back from microservice to monolith. When we searched alternative spellings and synonyms for these two terms, we referred to previous research and literature and to common terminological usage in the field to ensure that our searches covered different expressions. For the term "microservice", we selected the following strings, as they include the potential alternatives and plural forms *"(microservice* OR micro-service* OR "micro service*")"*. For the term "monolith", we directly selected the string *"monolith*"* with the asterisk in order to include the plural forms and adjectives.

We considered the related terms and phrases in order to conduct a more comprehensive search of the articles on the phenomenon of "switching back" in the research field. We chose terms that covered different aspects of "return", " refactor", "migrate", etc., to ensure that the search strings covered a wide range of descriptions and thus provided as comprehensive a search as possible *"(back OR return* OR refactor* OR rearchitect* OR migrat* OR revert* OR re-architect* OR redesign* OR reimplementation*)"*. Finally, we used boolean operators to relate the identified various terms. We used the OR operator to connect alternative spellings and synonyms and the AND operator to connect the main terms. The following search string represents the outcome:

"(microservice OR micro-service* OR "micro service*")"*
AND
*monolith**
AND
"(back OR return OR refactor* OR rearchitect* OR migrat* OR revert* OR re-architect* OR redesign* OR reimplementation*)"*

This search query was applied to both white and grey literature reviews.

2.2.1. White Literature

Sources. After identifying the search string, we decided on the literature databases to be searched for the white literature. We took SCOPUS (the SCOPUS database: https://www.scopus.com (accessed on 1 March 2024)), which is the most extensive literature database, as our main target. We also considered IEEEXPLORE (the IEEEXPLORE database: https://ieeexplore.ieee.org/ (accessed on 1 March 2024)), the ACM DIGITAL LIBRARY (the ACM DIGITAL LIBRARY: https://dl.acm.org (accessed on 1 March 2024)), and the citation database WEB OF SCIENCE (the WEB OF SCIENCE: https://www.webofscience.com/ (accessed on 1 March 2024)) to double-check the results obtained from SCOPUS. The main reasons for selecting

these databases were their potential completeness and popularity. The vast majority of publishers publish index papers on the selected databases, and they are widely recognized as the most representative of the research in software engineering [15], in addition to being used by many systems literature and mapping studies [16–18]. Additionally, to avoid analyzing papers that were not peer-reviewed, we decided not to adopt GOOGLE SCHOLAR (GOOGLE SCHOLAR: https://scholar.google.com (accessed on 1 March 2024)) as a database. This database includes preprints obtained from open-access archives like ARXIV, which are unpublished research.

Inclusion and Exclusion Criteria. We needed to screen the retrieved papers for consistency and adequacy to help address our research questions. Table 1 shows the complete list of inclusion and exclusion criteria.

The inclusion criteria were mainly relevant to whether the searched articles could address our research questions. We considered the defined research questions and determined that the change of switching back from microservice to monolith was the core aspect we were trying to investigate. If the articles mentioned this change, the content should relate to the reasons, processes, or practitioner opinions involving switching back. As described in Table 1, we included papers based on this inclusion criterion.

For the exclusion criteria, we first filtered out papers that were duplicated, not written in English, or unrelated to our research questions. We also excluded non-peer-reviewed papers. Even though we did not choose Google Scholar as a database, we could still obtain non-peer-reviewed articles. It is worth remembering that the change we needed to research was switching back from microservice to monolith, and it is important to get relevant industrial cases or other factual evidence. Therefore, we excluded papers unrelated to the related industrial cases or other factual evidence. In addition, if we found that a paper was an extension of a previously published paper, e.g., a journal paper was an extension of a conference paper, we retained only the extension and filtered out the previous preliminary version. Finally, we excluded papers that were not accessible by the institution.

Table 1. Inclusion and exclusion criteria.

Inc./Exc.	Criteria
Inclusion	Papers proposing "change from Microservice back to Monolith"
Exclusion	Not in English Duplicated Out of topic Not peer-reviewed papers Not referred to industrial cases or other factual evidence Not accessible

Search and Selection Process. After defining the search string, bibliographic sources, and inclusion and exclusion criteria, we started searching for the strings in the selected search databases. We used the "Advanced Search" option for each of the four databases selected. We mainly restricted the search within the search string to the title and abstract during the advanced search process. The purpose was to ensure that our search results more accurately reflected the literature relevant to our research topic while reducing the interference of irrelevant literature, thereby ensuring the accuracy and reliability of our research. The search was conducted and included all the publications available until the period. This process may have led to some related studies not being included, but the "snowballing" step will compensate for this situation. The search results are reported in Table 2. The initial search results comprised 571 papers, which was eventually reduced to 295 after removing duplicates. As shown in Figure 1, we continued with the steps in the search and selection process.

Table 2. Initial literature search by library.

Library	White Lit.
Scopus	277
IEEE	108
ACM	34
Web of Science	152
Non-duplicates	295

Applicability testing of inclusion and exclusion criteria: We first conducted applicability testing to verify the validity of the inclusion and exclusion criteria [19]. We tested on a subset of 20 papers (assigned to two authors) randomly selected from the papers retrieved. Authors met to discuss the results after the testing. This step did not ultimately result in revisions to the inclusion and exclusion criteria, probably indicating the completeness and suitability.

Applying inclusion and exclusion criteria to title and abstract: After ensuring the inclusion and exclusion criteria through the applicability testing, we applied the final inclusion and exclusion criteria to the retrieved papers. Two authors read each paper; in the case of disagreement, a third author was involved in the discussion to clear up any such disagreement. Out of the 295 initial papers, we selected three papers that were included based on the title and abstract. To evaluate the inter-rater agreement before involving the third author, we calculated Cohen's kappa coefficient [20]. Cohen's K coefficient scored 0.93, which represents the application of the inclusion and exclusion criteria, resulting in an almost perfect agreement.

Full reading: We read three full papers included by title and abstract and assigned each paper to two authors using the same criteria in Table 1. We engaged a third author for these papers to make a final decision. On this basis, we selected only one paper as a possibly relevant contribution.

Snowballing: We used the snowballing approach [21]. We considered all the references in the selected articles and evaluated all papers that cited the retrieved ones. We relied on GOOGLE SCHOLAR, which allows authors to search the reference papers easily to check whether some related studies were missing. Finally, the result showed there were no additional relevant papers.

Ultimately, only one paper was selected from the search and selection process among the 295 papers in the white literature review.

2.2.2. Grey Literature

Sources. We applied the same search string in the Google search engine, as it is the one that has been commonly used for grey literature searches in academia. In addition, we also searched for potential discussion threads on some popular tech-involved social media platforms, including Reddit, Quora, and StackOverflow.

Inclusion and Exclusion Criteria. We applied the same inclusion criteria shown in Table 1. Regarding the exclusion criteria, as this part is a grey literature review, we removed the "Not peer-reviewed papers" criterion. We also removed the criterion "Not in English" because within Google advanced search, we selected English as the language. The other exclusion criteria were the same.

Search and Selection Process. We used Google Advanced Search to search the results. We put search strings into the appropriate search boxes as required and selected the language as English to narrow the results. The search was conducted with 198 results returned in Google. Four forum discussion threads were found on Reddit and Quora each, while none were on StackOverflow. We selected 22 articles from the 198 results by directly full-reading the articles. Of the eight results from forum discussion threads, none provided helpful information. We also conducted snowballing for the 22 articles selected previously

by searching specifically for the company cases presented therein. We obtained nine extra articles, which represented 31 articles that were selected from the grey literature.

2.3. Data Extraction

From the one academic paper and 31 articles selected, we extracted the data and mapped them to answer each research question. Herein, we first extracted the categories that directly answered the RQs given by any of the selected articles. As different articles give different categories, we merged them by checking the potential direct mapping and then taking the union of the two sets. Based on the iterative discussion, the two authors have to agree on the final extracted data and the categories.

2.4. Replicability

We prepared a replication package (https://figshare.com/articles/dataset/MS2MO_MultiVocal/24999218 (accessed on 1 March 2024)) to enable other researchers to read and extend our study, and this package presents the MLR process methodology and the complete results obtained for this study.

3. Results

We selected 32 articles (white and grey) via the previously described process. When classifying these selected articles by year of publication, we can observe the research trend of switching back from microservice to monolith. As shown in Figure 2, the cumulative number of articles published has been increasing. These sources were first published in 2018, and their numbers increased slightly in 2020. The number of publications was stable in 2021 and 2022 but significantly increased in 2023.

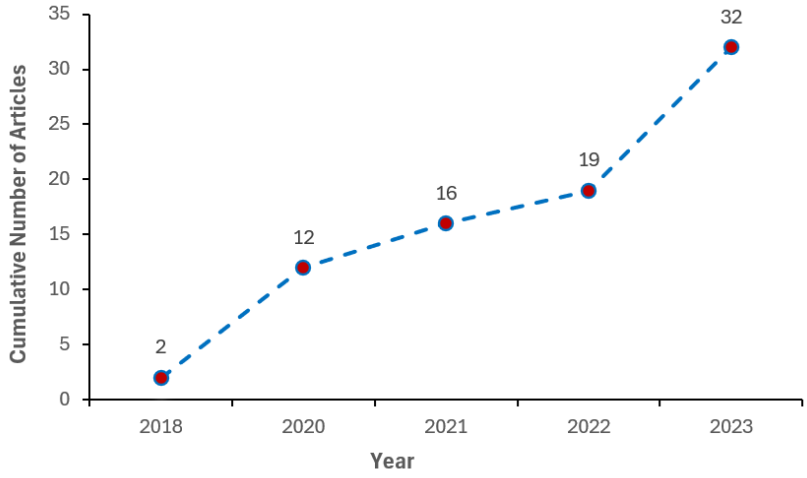

Figure 2. Cumulative number of articles over the years.

According to the review and analysis of the selected articles, there were four cases of switching back from microservice to monolith:

- Istio control plane [22]
- Amazon Prime Video monitoring service [12]
- Segment [23]
- InVision [24]

These articles were first published in 2018 and initially attracted little attention. With the occurrence of the Segment case in 2020, some discussion was generated among practitioners. The Amazon Prime Video monitoring service case in 2023 brought more discussion about switching back from microservices to monolith. Among them, the Amazon Prime video monitoring service has attracted the most attention from practitioners, with 10 articles.

3.1. RQ1: What Are the Reasons for Switching BACK from Microservice to Monolith?

From the four cases, we identified five main reasons for switching back from microservice to monolith, as reported in Table 3.

Table 3. Reasons for cases switch back to monolith.

Reason	Case	Selected Articles
Cost	Istio control plane, Prime Video monitoring service, Segment, InVision	[12,22,24–39]
Complexity	Istio control plane, Segment, InVision	[22–27,36,37,40,41]
Scalability	Istio control plane, Prime Video monitoring service, Segment	[12,22,23,25–29,32,33,35,39,42,43]
Performance	Segment	[23,39,40,43]
Organization	Istio control plane, InVision	[22,24,36,37]

Cost is the most common reason companies switch back from microservice to monolith, and Table 3 presents all four cases in our review involving cost reasons. In the Istio control plane, the system incurred a marginal cost due to the independently scalable components because of the microservice architecture, resulting in very little value [22,25,27]. Also, it had the problem of high operational costs because of the complexity of the microservice architecture [26]. Compared to the other three cases, the Prime Video monitoring service faced the most serious cost problem. It is a video streaming platform owned by Amazon. Initially, it used distributed components coordinated within AWS step functions, resulting in a total cost for all the artifacts that were not accepted for solutions on a broad scale, and the approach used for transferring video frames (images) between different modules was costly for numerous Tier-1 calls to S3 buckets. [12,28–35]. The cost-effectiveness of Prime Video monitoring service's switching back is also the most obvious. Amazon Prime Video claimed it reduced infrastructure costs by over 90% by migrating services from microservices back to monolithic architecture [12]. Segment faced significant cost challenges as well. Its operational costs associated with implementing microservices were unaffordable [38,39]. Regarding the InVision case, it commented that *"Microservices Furthermore, Have a Dollars-And-Cents Cost"* [24,36]. Its service operated on the server, interfacing with the database, reporting metrics, and generating log entries. Each activity incurred *"a very real dollar and cent cost"* [24]. Consequently, microservices incur expenses, particularly when taking into account the maintenance of redundancy to establish a highly available system [24,37].

Complexity is one of the most important reasons that companies switch back from microservice to monolith. Especially in the Istio control plane, we found that microservice architecture led to complexity in its content, management, and technology. First, its different planes used different languages to program [22]. Second, different teams worked on different modules individually. However, this approach introduced additional complexity and negatively affected user usability rather than allowing the development teams to manage them more conveniently [22,26]. In addition, all of the components within Istio's control plane were simultaneously consistently released in unified versions. At the same time, the decoupled versioning capabilities of microservices added a layer of complexity to the overall system [22]. Finally, the Istio control plane had limited isolation capabilities, and there was no guarantee that problems or changes in one microservice would not impact other microservices, so complete microservice isolation was difficult to achieve [22,25,27].

Furthermore, we found other factors in the Segment case that lead to higher complexity: managing multiple repositories and diverging shared libraries [23,40,41]. Initially, every module was classified as a different service, but they used one repository, which led to low efficiency. One corrupted test in the repository would affect all services, and deployment changes required fixing irrelevant tests [23,40,41]. Separating the code for every module into different repositories added more complexity and maintenance workload. Segment built shared libraries to achieve common transformations and functions in order

to reduce the burden of the development and maintenance of these code libraries [23]. InVision had similar complexity issues as Segment. InVision grew increasingly because of microservice architecture, having more repositories, programming languages, databases, monitoring dashboards, etc., causing significant development and maintenance pressure and complexity on the development team [24,36,37].

Scalability. The feature of microservices being able to extend components independently is often a key reason companies decide to migrate to the microservices architecture. However, in these cases, this benefit turned into a drawback. In Istio, the respective function (XDS) decided the cost of the control plane [22,25,27]. In contrast, these functions had marginal costs, making the isolation of microservices of little value [22,26]. Faced with this, the Istio control plane switched back to monolith instead, making it much easier to scale. Amazon Prime Video monitoring service also encountered a scaling limitation. It met a hard scaling limit with around 5% of the expected load [12,28,29,32,33,35,42], and this was due to the implementation of orchestration management by AWS step functions [12,28,29,35,42]. Unlike Amazon Prime Video monitoring service, Segment's scaling challenge was the reason for the automated scaling configuration. The management and scalability of each microservice became a crucial operational expense due to the increased number of services [23]. Each service has specific load patterns that require manual scaling to handle unplanned spikes [43]. Adjusting automatic and scalable configurations became more challenging because of the various resourcing requirements of each service [23,39].

Performance. Only Segment mentioned performance among the reasons for switching back to the monolith out of these four cases, and it was one of the most significant reasons for Segment switching back. The worst problem was the line head blockage. The microservice architecture caused head-of-line blocking, where a failure in one destination causes delays in all destinations [23,40]. This affected the timeliness of event delivery and also customer satisfaction [43]. *"The team soon realized that the channel was blocked, causing performance issues when the tool returned an error. It was why they decided to move out of monolith into all the microservices"*, said Alexandra Noonan, the software engineer who led the project in her blog [39]. Furthermore, high complexity was also a factor that caused decreased system performance. The high complexity of microservices brought development teams to a standstill, and the benefits of modularity and autonomy changed to strain, which delayed them and reduced efficiency, leading to reduced performance [23].

Organization. The high complexity of team management is also one of the key reasons for the companies' switching back to monolith, especially the Istio control plane and InVision. In the Istio control plane, microservices enable individual services' management by separate teams, but in fact, this could create confusion for development teams trying to manage more efficiently [22]. In comparison, InVision had more serious team issues. InVision started with a small legacy team but owned many repositories, databases, programming languages, etc. [24]. Over time, Conway's Law benefits turned into a strain on the legacy team as the scale was inappropriate, so switching the microservices back to monolith became necessary [24,36,37].

3.2. RQ2: What Are the Key Aspects to Pay Attention to during the Switching Back Process?

We analyzed six key aspects that should be taken into account during the process of switching back from microservice to monolith, combined with four cases from the company and some other thoughts from practitioners.

Stop developing more services. This implies that it cannot import new microservices. Migrating from microservices to monolith requires use of the existing architecture as the "center" for the future monolith to carry the new features [44]. All of the services would eventually need to be integrated into this center. However, if we continue developing new services after we have decided to move back to the monolithic architecture, the system may become confused.

Consolidate and test paths. Microservices may form a singular and consistent process between several systems. Consolidation paths are essential as systems are integrated from

microservices to monoliths [44]. After the migration, testing the path is also important in order to ensure the system runs well. These processes help the reversed monolithic system function properly and satisfy all requirements. [45].

Unify data storage. Data storage is one of the most important aspects of switching back to a monolith. Shanea thought we had two possible choices: separating data or transferring data to one individual database [45]. The former allows for maintaining the independence and isolation of individual components and could hold to a more homogeneous structure. The latter allows for lower costs, higher performance, and reduced system complexity [45]. The company's development team should select the appropriate data storage approach according to the assessment of the actual situation. Otherwise, it will cause a great deal of trouble.

Implement the message bus principle. Implementing a message bus like Kafka can act as an indirect layer during the transition [45]. This approach allows for the progressive merging from microservices to monoliths without disrupting existing systems. The message bus enables smooth communication between components, allowing for the easy decomposition and reconstruction of services as needed.

Give up diverse techniques. Different services in microservices can use different programming languages and frameworks. However, this diversity would require changing after the system switched back. For instance, the majority of systems should use a maximum of two back-end languages simultaneously [44]. If the diversity of technologies is retained, switching back to monolith cannot be completed.

Learn to use modular design principles. Even when switching back from microservice to monolith, we need to maintain a modular design. Modular design enables code organization as different clearly bounded modules, promoting concerns' separation and maintainability [44]. Moreover, the modular design also gives the system advantages in terms of both the flexibility and modularity of microservices as well as the simplicity and usability of monoliths [45].

3.3. RQ3: What Are the Opinions of the Other Practitioners Regarding Such Switching Back?

Other practitioners had mixed opinions about this switching-back behavior. Some argued that this way is correct. They thought that microservice was not the "utopian application architecture" [25]. David Heinemeier Hansson scoffed that microservice was a zombie architecture [29]. Some practitioners thought monoliths had an advantage over microservices because they are easier to code, scale, deploy, test, and use to deal with cross-domain issues [46,47]. In contrast, some did not agree with the practice of switching back. They believe that microservices are still one of the most popular architectures. Angel posted that monoliths are not the solution, and organizations need to think better and support proactive communication channels that can supply the gaps between teams [48]. However, the vast majority of practitioners still believed that the idea of switching back to a monolithic architecture requires consideration of the actual system's situation and principles, and that switching back calls for an assessment of whether monolith is really the best fit for the company's requirements, such as team size, structure, skills, and operational abilities [45,49]. Arnold Galovics stated "Do not start with microservices in production", and proposed that systems cannot start with microservice or blindly move to microservice for the sake of its popularity [50]. He suggested that people should never blindly choose microservices over monolith because moving from monoliths to microservices is much easier than switching back [50]. Moreover, most of the disadvantages of monoliths and microservices are well-known [51], so Itiel believed that the architecture recommendation depends on the project type [52].

4. Discussion

This section first discusses analysis of the results as related to the research questions within the multivocal literature review and subsequent thoughts. Then, we discuss the

implications of switching back and the significance of this study. Finally, we introduce our study limitations and possible future work.

4.1. Interconnected Relations and Thoughts between RQs

The results of RQ_1 and RQ_3 indicate that some companies and researchers have discovered that the microservices architecture is not always the best choice, and that the choice of architecture should consider the product's actual situation rather than blindly following the trend. As summarized in RQ_1, the characteristics of the microservices architecture may lead to problems like cost, complexity, organizational structure, etc. On the other hand, changing business requirements may lead companies to need more tightly integrated systems like monolithic architecture.

Combined with the results of RQ_1 and RQ_2, we found that economic considerations were the primary motivating factor for microservices switching back to monolith. The economic implications of microservices, as highlighted in RQ_1, coupled with the emphasis on stopping the development of new microservices as mentioned in RQ_2, emerged as the decisive factors influencing organizations to reevaluate their architectural choices. Additionally, the complexity introduced by microservices, both in terms of content and management, is intertwined with scalability, forming a complex network of factors (RQ_1). The need for consolidating paths and unifying data storage during the switch-back process (RQ_2) responds to these issues.

The convergence of practitioners' views (RQ_3) provided a rich network of perspectives on our research goal and supplemented the first two RQ_s with information of a broader discussion. It demonstrated the diversified views of practitioners on whether to switch from microservices back to the monolith. On the one hand, some practitioners expressed doubts about microservices, while others advocated for a monolithic architecture, emphasizing its simplicity and ease of use. However, most practitioners still believe that the benefits of switching back to monolith are not absolute and that the decision to do so should be based on "situational considerations". These three views present a dialectic between microservices and monolith, highlighting the complexity practitioners face in understanding and evaluating architectural decisions. This also corresponds to the factors that (RQ_1 and RQ_2) organizations need to balance in practice.

Notably, we extracted aspects with RQ_2 to learn to use modular design guidelines, and this seems to be related to another concept: modular monolith. Of the four cases in RQ_1, no company explicitly stated that they were switching from microservices back to the modular monolith, just the monolith. However, some practitioners in RQ_2 did make the case in their articles to keep the modular design because it can take advantage of both monoliths and microservices. This seems to be a signal that practitioners are starting to take notice of the concept of modular design or modular monoliths, but there is no case for any of them at the moment.

4.2. Implications and Significance

We discussed the implications to the system architecture and companies of systems switching back from microservice to monolith, combining the results of the three research questions. In terms of system architecture, this switching back leads to overall system complexity and performance changes, increasing the short-term development and maintenance costs. For the company, since microservices and monolithic architectures do not have the same development methodology, switching back from microservice to monolith may require the company to reconfigure its development teams to adapt to the development requirements and processes of the new architecture. This may involve adjustments in team size, changes in the technology stack, and the division of responsibilities within the team. In addition, the monolithic architecture may require more centralized development and maintenance, so the company's departmental decision-making and collaboration would need to become ore centralized. Finally, different architectural choices reflect different organizational values and priorities. This may require adjustments to the organizational

culture to accommodate the changes brought about by the new architectural choices. This switching back will, in the short term, cause a certain amount of consumption, burden, and change to the cost, complexity, organizational structure, and cultural factors of the company, indicating the necessity for an assessment of whether the monolithic structure is suitable for the company's actual situation before the switching back.

The significance of the study is three-fold. This study provides a comprehensive view of the reasons, actions, and perspectives of organizations and practitioners as to switching back from microservice to monolith, demonstrating to researchers and companies that the microservices architecture may not always be the best choice, and instead, that the selection between monolithic and microservices architectures should be based on the specific needs of the company and its systems through assessment. In addition, the study provides a reference for companies switching back from microservice to monolith. Finally, the study provides researchers with the actions and opinions of organizations and practitioners as to switching back from microservice to monolith, deepening the connection between academia and industry and helping researchers in further related research fields.

4.3. Study Limitations

The study collected and analyzed data through the multivocal literature review methodology, and it did not further investigate the perceptions of industry practitioners on this topic through survey or interview methods. For reasons like confidentiality policies, practitioners usually do not publish the latest case content related to their companies on the Internet, leading to grey literature review results that are not the latest or are just opinions without concrete analysis of specific cases.

The key aspects to pay attention to during the switching back process in response to RQ_2 are based on existing cases and practitioners' opinions in the selected articles. However, different companies have different actual situations, so these steps can only be used on a reference basis and cannot be directly replicated. Moreover, some articles also proposed the design of monolith "modularisation". This indicates that the monolith they switched back to may not be the traditional monolithic architecture used in the past but rather the "modular monolith". However, the articles did not discuss more about modular monoliths or whether the converted monoliths and traditional monolithic architectures are different. Some researchers have discussed whether modular monolith is the new trend in software architecture [53], but there are no concrete actual cases to support it. On the other hand, it is worth noting that articles did not present the shortcomings after companies switched back to the monolithic architecture, only reported the improvements and benefits. Regardless of the choice of architecture, there is a risk of a suboptimal implementation of the information system.

The results of RQ_3 collected practitioners' opinions regarding switching back from microservice to monolith. However, these opinions may have bias. Given that practitioners are in the industry, their opinions are subjective and may be influenced by their companies or surroundings. This also explains why the opinions of practitioners in RQ_3 were mixed.

4.4. Future Work

Our future research will start by further investigating the perceptions of industry practitioners on this topic. Through survey and interview methods, we can better understand the latest cases or opinions regarding switching back from microservice to monolith through the lens of evolving industry trends and challenges.

We will also conduct comparative case studies on the performance of microservice and reversed monolith systems. By comparing the cost efficiency, scalability, response time, fault tolerance, resource utilization, deployment, and maintenance complexity between the both, we can better understand the strengths and weaknesses of microservices and reversed monolithic systems in different scenarios and provide valuable insights.

Moreover, we will research the application and impact of modular monoliths in the industry. Modular monolith has become popular in the software architecture field in recent

years [53], and the questions of whether it is the new trend in software architecture, the similarities and differences with traditional architecture and microservices, its application in industry, and the relationship with the traditional monolith in switching back will be the future directions of our research, helping researchers to understand the new trend in software architecture.

5. Threats to Validity

In this section, there is a discussion of the potential threats to the validity of this study. We follow the guidelines proposed by Ampatzoglou et al. [54], and compared to the guidelines by Wohlin et al. [55], these are particularly applicable to secondary research in the field of software engineering. Herein, we discuss the study selection validity, data validity, and research validity of our study.

5.1. Study Selection Validity

In this study, we strictly followed the guidelines for conducting multivocal literature reviews in software engineering proposed by Garousi et al. [14] regarding the search process steps, source selection, data extraction, and data synthesis. With this guideline, the threat to the research process is significantly reduced. During the initial search, we identified all keywords carefully and diversified them using synonyms. Although this search method can cover most publications, there are still issues and limitations that inevitably arise, such as the fact that the search is limited to titles and abstracts. We also utilized a grey literature search. To extend the coverage of our study, especially in the grey literature search, we used "snowballing" as a complementary approach. We reviewed all the references in the selected studies and other articles involved in the grey literature, which resulted in nine relevant articles being found.

Regarding the formulation of inclusion and exclusion criteria, we also followed the guidelines proposed by Petersen et al. [18]. All authors discussed these criteria and fit in with the topic of our study. Both authors independently conducted the research selection in the "apply criteria to title and abstract" and "full reading" steps. In case of disagreement, the third author would participate in the discussion in order to reach a conclusion. Meanwhile, when the study was in the first round of revision in 2024, we conducted the white and grey literature search again, and additional studies generated by the new query were added to the included articles, which reduced the potential threat of an incomplete paper.

5.2. Data Validity

During the data extraction process, two authors contributed independently. The process was iterative, especially for the grey literature. We identified the categorization through an open coding method. In RQ1, we first categorized the articles based on the company whose case was discussed therein and then categorized them a second time based on that company's reasons for switching back. In the data synthesis process, we used the descriptive (narrative) synthesis method based on the guideline paper for synthesizing evidence in SE research [56]. As this study was not a systematic mapping study, we categorized the extracted results and summarized them narratively. In addition, we only selected publicly available academic papers and grey articles due to confidentiality policies, which prevented us from including more relevant content from the industry. As noted above, in future work, we plan to conduct industry surveys to learn more about industry views on this issue, thereby reducing the threat to validity.

5.3. Research Validity

The research method was determined by all authors after several discussions before the study began. All authors agreed on the decision to use a multivariate literature review, which reduced the threat of methodological bias in the study. While identifying the research questions, the authors also had an iterative discussion because of some differences in opinion, and finally, they identified the research questions together. In addition, the study

can be replicated by strictly following the details of the research process. The methodology step also describes the details of the search string and multivocal literature review. With future technological advances, such replication will likely be applied to studying software architectures for the next generation.

6. Related Work

Many approaches have been proposed for switching from monolith to microservice to address decomposition, refactoring, scaling, and many other issues during the process. Chen et al. [57] proposed a top-down analysis approach and developed a dataflow-driven microservice decomposition algorithm. The algorithm can provide reasonable, objective, and understandable microservice candidates through strict and practical implementation procedures. On the other hand, Taibi and Systä [7] proposed a process mining-based decomposition framework to reduce the subjectivity of the decomposition process, which can improve the overall system decomposition quality. In addition, Eski and Buzluca [58] proposed a new approach for converting existing applications into microservices using code repositories. This approach uses static code coupling and graph clustering methods to automatically extract microservices from monoliths, achieving an 89% success rate. Fritzsch et al. [59] classified refactoring approaches to migrating from monolithic systems to microservices. They found that most approaches are limited in their use by various specific conditions and require adequate tool support. Furthermore, Abbott and Fischer [60] proposed another decomposition based on scalability, where applications are partitioned into smaller components to achieve higher scalability.

With more companies migrating from monolithic systems to microservices, the drawbacks and challenges of microservice architectures are magnified. Soldani et al. [8] summarized the pain of microservices mainly due to the inherent complexity of microservices based on a systematic grey literature review. For instance, after migrating from monoliths to microservices, the system's large number of microservices and parallel dependencies make it unbearable for the development teams. Taibi and Lenarduzzi [61] presented bad practices regarding microservices derived from interviews with developers using microservices-based systems.

There are also other studies researching the challenges of microservices architecture. Wang et al. [62] supposed that the design and implementation of microservices are performed in a defined IoT environment, but besides extending behaviors to the system and extending the problems it solves, more features and functionalities need to be integrated into the microservices architecture, and the maintenance of such an architecture can become a serious problem or challenge. Driss et al. [63] proposed that microservice systems may be under more security attacks, defined as network, software, and data attacks, than monoliths. They concluded that because microservice containers are highly replicable, a vulnerability in one module could quickly escalate into a more significant problem. Moreover, Atitallah et al. [64] reported that the independence of microservices can challenge organizational culture as developers do not have the same global visibility as they would in a monolithic architecture, and the inherent isolation between teams can lead to challenges in communication and decision-making. However, no studies are specifically focused on the phenomenon of switching back from microservice to monolith.

In summary, we perceive that the phenomenon of switching back from microservice to monolith may stem from unsuitability and the pains companies feel after migrating from monoliths to microservices. The key motivation of our study is to fill this gap, investigate the reasons why companies decide to switch back to monolith, and identify key aspects to pay attention to during the process. At the same time, opinions from other practitioners are also a focus of the study.

7. Conclusions

There are ongoing discussions regarding switching back from microservice to monolith, and some companies have already taken action. Though it is still too early to claim it as

a trend, the existing cases and researchers' and practitioners' opinions are worth noticing. We wanted to understand the reasons and methods for switching back from microservice to monolith as well as practitioners' opinions and advice towards such practices. We conducted a multivocal literature review, preliminarily investigating why companies decided to switch back to the monolith and critical aspects to pay attention to during the process. At the same time, we analyzed other practitioners' opinions regarding this phenomenon. We selected 32 articles (white and grey) via the multivocal literature review. The summarized results show that cost is the most important reason companies switch from microservice back to monolith. Furthermore, complexity, scalability, performance, and organization are key reasons for this trend. During the process of switching back, we determined that there are six key aspects worth noticing: (1) stopping the development of more services, (2) consolidating and testing paths, (3) unifying data storage, (4) implementing the message bus principle, (5) giving up diverse techniques, and (6) learning to use modular design principles. As to the practitioners' opinions, they had mixed views about the switching back phenomenon, but most thought that switching back required consideration of the actual system situation and principles. This study provides insights to researchers and practitioners, especially that the choice of architecture should be made through assessment based on the company's specific needs and its systems; provides references for companies switching back from microservice to monolith; and helps researchers in related research fields.

Author Contributions: Conceptualization, R.S., X.L. and D.T.; data curation, R.S.; formal analysis, R.S.; methodology, R.S. and X.L.; investigation, R.S. and X.L.; validation, R.S.; writing—original draft preparation, R.S.; writing—review and editing, R.S, X.L. and D.T.; visualization, R.S.; supervision, D.T.; project administration, R.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Research Council of Finland (grant 349488—MuFano) and Business Finland (grant 24304494—6GSoft).

Data Availability Statement: The data presented in this study are openly available in FigShare at https://figshare.com/articles/dataset/MS2MO_MultiVocal/24999218, accessed on 15 January 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Esparza-Peidro, J.; Muñoz-Escoí, F.D.; Bernabéu-Aubán, J.M. Modeling microservice architectures. *J. Syst. Softw.* **2024**, *213*, 112041 . [CrossRef]
2. Taibi, D.; Lenarduzzi, V.; Pahl, C. Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. *IEEE Cloud Comput.* **2017**, *4*, 22–32. [CrossRef]
3. Amoroso d'Aragona, D.; Li, X.; Cerny, T.; Janes, A.; Lenarduzzi, V.; Taibi, D. One microservice per developer: Is this the trend in OSS? In Proceedings of the European Conference on Service-Oriented and Cloud Computing, Wittenberg, Germany, 22–24 March 2023; pp. 19–34.
4. Osman, M.H.; Saadbouh, C.; Sharif, K.Y.; Admodisastro, N. From Monolith to Microservices: A Semi-Automated Approach for Legacy to Modern Architecture Transition using Static Analysis. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 907–916. [CrossRef]
5. Taibi, D.; Lenarduzzi, V.; Pahl, C. Architectural patterns for microservices: A systematic mapping study. In Proceedings of the CLOSER 2018: 8th International Conference on Cloud Computing and Services Science, Funchal, Portugal, 19–21 March 2018.
6. Lenarduzzi, V.; Lomio, F.; Saarimäki, N.; Taibi, D. Does migrating a monolithic system to microservices decrease the technical debt? *J. Syst. Softw.* **2020**, *169*, 110710. [CrossRef]
7. Taibi, D.; Systä, K. From monolithic systems to microservices: A decomposition framework based on process mining. In Proceedings of the International Conference on Cloud Computing and Service Science—CLOSER 2019, Crete, Greece, 2–4 May 2019.
8. Soldani, J.; Tamburri, D.A.; Van Den Heuvel, W.J. The pains and gains of microservices: A systematic grey literature review. *J. Syst. Softw.* **2018**, *146*, 215–232. [CrossRef]
9. Lewis, J.; Fowler, M. Microservices: A Definition of This New Architectural Term. 2014. Available online: https://martinfowler.com/articles/microservices.html (accessed on 1 March 2024).
10. Dragoni, N.; Giallorenzo, S.; Lafuente, A.L.; Mazzara, M.; Montesi, F.; Mustafin, R.; Safina, L. Microservices: Yesterday, today, and tomorrow. *Present Ulterior Softw. Eng.* **2017**, 195–216. [CrossRef]
11. Jamshidi, P.; Pahl, C.; Mendonça, N.C.; Lewis, J.; Tilkov, S. Microservices: The journey so far and challenges ahead. *IEEE Softw.* **2018**, *35*, 24–35. [CrossRef]

12. Kolny, M. Scaling up the Prime Video Audio/Video Monitoring Service and Reducing Costs by 90%. 2023. Available online: https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90 (accessed on 1 March 2024).
13. Gos, K.; Zabierowski, W. The comparison of microservice and monolithic architecture. In Proceedings of the 2020 IEEE 16th International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), Polyana, Ukraine, 12–16 May 2021; pp. 150–153.
14. Garousi, V.; Felderer, M.; Mäntylä, M.V. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Inf. Softw. Technol.* **2019**, *106*, 101–121. [CrossRef]
15. Kitchenham, B.; Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. 2007. Available online: https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf (accessed on 1 March 2024).
16. Azeem, M.I.; Palomba, F.; Shi, L.; Wang, Q. Machine learning techniques for code smell detection: A systematic literature review and meta-analysis. *Inf. Softw. Technol.* **2019**, *108*, 115–138. [CrossRef]
17. Hall, T.; Beecham, S.; Bowes, D.; Gray, D.; Counsell, S. A systematic literature review on fault prediction performance in software engineering. *IEEE Trans. Softw. Eng.* **2011**, *38*, 1276–1304. [CrossRef]
18. Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12, Bari, Italy, 26–27 June 2008; pp. 1–10.
19. Kitchenham, B.; Brereton, P. A systematic review of systematic review process research in software engineering. *Inf. Softw. Technol.* **2013**, *55*, 2049–2075. [CrossRef]
20. Landis, J.R.; Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics* **1977**, *33*, 159–174. [CrossRef] [PubMed]
21. Wohlin, C. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In Proceedings of the EASE 2014, London, UK, 13–14 May 2014.
22. Box, C. Introducing Istiod: Simplifying the Control Plane. 2020. Available online: https://istio.io/latest/blog/2020/istiod/ (accessed on 1 March 2024).
23. Noonan, A. Goodbye Microservices: From 100s of Problem Children to 1 Superstar. 2018. Available online: https://segment.com/blog/goodbye-microservices/ (accessed on 1 March 2024).
24. Nadel, B. Why I Have Been Merging Microservices Back into the Monolith at InVision. 2020. Available online: https://www.bennadel.com/blog/3944-why-ive-been-merging-microservices-back-into-the-monolith-at-invision.htm (accessed on 1 March 2024).
25. Posta, C. Istio as an Example of When Not to Do Microservices. 2020. Available online: https://blog.christianposta.com/microservices/istio-as-an-example-of-when-not-to-do-microservices/ (accessed on 1 March 2024).
26. Bianchi, L. Istio Is Moving Back to the Monolith, But It Doesn't Mean You Have to Do the Same. 2020. Available online: https://aletheia.medium.com/istio-back-to-monolith-and-you-88dd3bd23265 (accessed on 1 March 2024).
27. Mendonça, N.C.; Box, C.; Manolache, C.; Ryan, L. The monolith strikes back: Why istio migrated from microservices to a monolithic architecture. *IEEE Softw.* **2021**, *38*, 17–22. [CrossRef]
28. Jackson, J. Return of the Monolith: Amazon Dumps Microservices for Video Monitoring. 2023. Available online: https://thenewstack.io/return-of-the-monolith-amazon-dumps-microservices-for-video-monitoring/ (accessed on 1 March 2024).
29. Jennings, R. Microservices Sucks—Amazon Goes Back to Basics. 2023. Available online: https://devops.com/microservices-amazon-monolithic-richixbw/ (accessed on 1 March 2024).
30. Anderson, T. Reduce Costs by 90% by Moving from Microservices to Monolith: Amazon Internal Case Study Raises Eyebrows. 2023. Available online: https://devclass.com/2023/05/05/reduce-costs-by-90-by-moving-from-microservices-to-monolith-amazon-internal-case-study-raises-eyebrows/ (accessed on 1 March 2024).
31. Targett, E. Amazon Prime Video Team Throws AWS Serverless under a Bus. Available online: https://www.thestack.technology/amazon-prime-video-microservices-monolith/ (accessed on 1 March 2024).
32. Jellick, C. Microservices are Dead, Long Live the Monolith. 2023. Available online: https://www.acorn.io/microservices-are-dead-long-live-the-monolith/ (accessed on 1 March 2024).
33. Weinberg, N. 6 Lessons from the Amazon Prime Video Serverless vs. Monolith Flap. Available online: https://www.networkworld.com/article/3697737/6-lessons-from-the-amazon-prime-video-serverless-vs-monolith-flap.html (accessed on 1 March 2024).
34. Mooter, D. The Death of Microservices? 2023. Available online: https://www.forrester.com/blogs/the-death-of-microservices/ (accessed on 1 March 2024).
35. Bhasin, A. Exploring Amazon Prime Video's Architecture: Migrating from Microservices to Monolith for Audio/Video Monitoring Service. 2023. Available online: https://medium.com/@anshita.bhasin/exploring-amazon-prime-videos-architecture-migrating-from-microservices-to-monolith-for-aacbf9fabc73 (accessed on 1 March 2024).
36. ForkMyBrain. Why I Have Been Merging Microservices Back into the Monolith at InVision. 2020. Available online: https://notes.nicolevanderhoeven.com/readwise/Articles/Why+I've+Been+Merging+Microservices+Back+Into+the+Monolith+at+InVision (accessed on 1 March 2024).
37. Rmason. I Have Been Merging Microservices Back into the Monolith. 2020. Available online: https://news.ycombinator.com/item?id=25498079 (accessed on 1 March 2024).

38. Ramakani, A. It's Back! A Trend to Build Monolith. Available online: https://itnext.io/its-back-a-trend-to-build-monolith-852aaa5e086f (accessed on 1 March 2024).
39. Adfolks. Transitioning from Monolithic to Microservices Architecture: Pros, Cons, and Segment's Journey. 2021. Available online: https://www.adfolks.com/blogs/transitioning-from-monolithic-to-microservices-architecture-pros-cons-and-segments-journey (accessed on 1 March 2024).
40. Betts, T. To Microservices and Back Again—Why Segment Went Back to a Monolith. 2020. Available online: https://www.infoq.com/news/2020/04/microservices-back-again/ (accessed on 1 March 2024).
41. Noonan, A. To Microservices and Back Again. 2020. Available online: https://www.infoq.com/presentations/microservices-monolith-antipatterns/ (accessed on 1 March 2024).
42. Danilov, D. My Thoughts on Microservices and Monoliths. 2023. Available online: https://ddanilov.me/thoughts-about-microservices (accessed on 1 March 2024).
43. Meyer, D. Segment Struggled with Microservices, Went Back to Monolith. 2018. Available online: https://www.sdxcentral.com/articles/news/segment-struggled-with-microservices-went-back-to-monolith/2018/08/ (accessed on 1 March 2024).
44. Heinemeier, D. How to Recover from Microservices. 2023. Available online: https://world.hey.com/dhh/how-to-recover-from-microservices-ce3803cc (accessed on 1 March 2024).
45. Leven, S. The Great "Un-Migration": Migrating from Microservices Back to a Monolith. 2023. Available online: https://learn.codesee.io/migrating-from-microservices-to-a-monolith/ (accessed on 1 March 2024).
46. Doglio, F. Microservices Aren't Always the Answer: A Case for Monoliths. 2022. Available online: https://blog.bitsrc.io/when-microservices-are-not-the-answer-a-case-for-monoliths-895ccefa2728 (accessed on 1 March 2024).
47. Almog, S. Is It Time to Go Back to the Monolith? 2023. Available online: https://dev.to/codenameone/is-it-time-to-go-back-to-the-monolith-3eok (accessed on 1 March 2024).
48. Paredes, A. Bring Back the Monolith. 2022. Available online: https://medium.com/glovo-engineering/bring-back-the-monolith-92de928ae322 (accessed on 1 March 2024).
49. O'Donnell, C. Monolith to Microservices, and Back Again? 2020. Available online: https://ciaranodonnell.dev/posts/the-microservices-monolith-pendulum/ (accessed on 1 March 2024).
50. Galovics, A. Don't Start with Microservices in Production—Monoliths Are Your Friend. 2021. Available online: https://arnoldgalovics.com/microservices-in-production/ (accessed on 1 March 2024).
51. Sogos, D.D. Microservices's Dark Side: The Monolith Strikes Back. 2020. Available online: https://www.hexacta.com/microservicess-dark-side-the-monolith-strikes-back/ (accessed on 1 March 2024).
52. Maayan, I. Will Modular Monolith Replace Microservices Architecture? 2022. Available online: https://medium.com/att-israel/will-modular-monolith-replace-microservices-architecture-a8356674e2ea (accessed on 1 March 2024).
53. Su, R.; Li, X. Modular Monolith: Is This the Trend in Software Architecture? *arXiv* **2024**, arXiv:2401.11867.
54. Ampatzoglou, A.; Bibi, S.; Avgeriou, P.; Verbeek, M.; Chatzigeorgiou, A. Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Inf. Softw. Technol.* **2019**, *106*, 201–230. [CrossRef]
55. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B. *Experimentation in Software Engineering*; Springer: Cham, Switzerland, 2012.
56. Cruzes, D.S.; Dybå, T. Synthesizing evidence in software engineering research. In Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, USA, 16–17 September 2010; pp. 1–10.
57. Chen, R.; Li, S.; Li, Z. From monolith to microservices: A dataflow-driven approach. In Proceedings of the 2017 24th Asia-Pacific Software Engineering Conference (APSEC), Nanjing, China, 4–8 December 2017; pp. 466–475.
58. Eski, S.; Buzluca, F. An automatic extraction approach: Transition to microservices architecture from monolithic application. In *Proceedings of the 19th International Conference on Agile Software Development: Companion*; ACM: New York, NY, USA, 2018; pp. 1–6.
59. Fritzsch, J.; Bogner, J.; Zimmermann, A.; Wagner, S. From monolith to microservices: A classification of refactoring approaches. In Proceedings of the Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, 5–6 March 2018; pp. 128–141.
60. Abbott, M.L.; Fisher, M.T. *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*; Addison-Wesley Professional: Boston, MA, USA, 2015.
61. Taibi, D.; Lenarduzzi, V. On the definition of microservice bad smells. *IEEE Softw.* **2018**, *35*, 56–62. [CrossRef]
62. Wang, Y.; Kadiyala, H.; Rubin, J. Promises and challenges of microservices: An exploratory study. *Empir. Softw. Eng.* **2021**, *26*, 63. [CrossRef]
63. Driss, M.; Hasan, D.; Boulila, W.; Ahmad, J. Microservices in IoT security: Current solutions, research challenges, and future directions. *Procedia Comput. Sci.* **2021**, *192*, 2385–2395. [CrossRef]
64. Atitallah, S.B.; Driss, M.; Ghzela, H.B. Microservices for data analytics in iot applications: Current solutions, open challenges, and future research directions. *Procedia Comput. Sci.* **2022**, *207*, 3938–3947. [CrossRef]