Exploring the Pros and Cons of Monolithic Applications versus Microservices

Daniel dos Santos Krug[a], Rafael Chanin[b] and Afonso Sales[c]
School of Technology, PUCRS, Porto Alegre, RS, Brazil

Keywords: Microservices Architecture, Monolithic Applications, Architectural Decision Making.

Abstract: Microservices architecture emerged as an alternative to monolith architecture. With monoliths, applications are developed in entire blocks that communicate internally, manage their data usually in a single database, and each new feature demands the deployment of the application as a whole. On the other hand, microservices splits the application into smaller blocks with unique responsibilities, using lightweight communication mechanisms and managing their own data. This new architecture has several advantages, but it also has some disadvantages. From the understanding of these advantages and disadvantages, the main goal of this research is to identify how the two architectures have been used in professional practices. As well as how the academy can help in the understanding that if the microservices architecture entails a longer development time for the applications, in order to understand when the decision for an architecture may be more appropriate with another in software development. To fulfill the explained objective, it is intended to carry out a systematic study with the snowballing technique, research in the grey literature and a survey with specialists.

1 INTRODUCTION

The microservices architecture decomposes traditional monolithic applications into smaller, autonomous services (Richardson, 2016). These services, possibly written in different languages, interact through APIs and are commonly managed in containers such as Docker and Kubernetes (Bernstein, 2014), facilitating deployment across varied environments.

Early academic studies on microservices date back to the mid-2010s (Viennot et al., 2015; Stubbs et al., 2015), with discussions on the term's origins remaining unclear. Microservices offer several advantages over monolithic architectures, such as faster delivery, improved scalability, and increased autonomy (Soldani et al., 2018). These benefits stem from practices as Domain Driven Design (Merson and Yoder, 2020) and its bounded contexts, promoting a modular structure ideal for larger development teams and independent deployment, while allowing for diverse technologies.

However, microservices also introduce challenges, including security concerns, as each service requires individual protection (Yarygina and Bagge, 2018), and network performance issues due to API-based communication (Liu et al., 2020). Data consistency is another problem, as each service manages its data, potentially leading to inconsistencies (Kholy and Fatatry, 2019).

These disadvantages are well-studied and can be mitigated during development. Nonetheless, the added complexity and cost of developing communication layers, ensuring security, and maintaining data consistency make the development process more demanding, which is the focus of this research.

It is crucial to note that this research does not discourage the use of microservices; when properly implemented, the architecture's advantages often make it the preferred choice for many applications.

The research foundation and examples highlight significant benefits of adopting a microservices architecture. However, it also reveals complexities not fully addressed by the scientific community, suggesting areas for further exploration.

Building on the aforementioned insights, and given the absence of literature specifically addressing the potential increase in software development time with microservices, this study proposes an exploratory investigation into whether this issue is significant. The study's goal is to elucidate the pros

[a] https://orcid.org/0009-0008-8984-7801
[b] https://orcid.org/0000-0002-6293-7419
[c] https://orcid.org/0000-0001-6962-3706

Krug, D., Chanin, R. and Sales, A.
Exploring the Pros and Cons of Monolithic Applications versus Microservices.
DOI: 10.5220/0012703300003690

In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 2*, pages 256-263
ISBN: 978-989-758-692-7; ISSN: 2184-4992

and cons of implementing monolithic or microservices architectures in software development, providing a balanced perspective to aid developers in making informed architectural decisions.

The main research question of this work is formulated as follows: *"How can developers practicing microservices architecture be supported in making decisions regarding its use or the choice of traditional architectures, such as monoliths?"*

The overarching goal of this research is to discern the conditions under which monolithic architecture is adequate for application development and when microservices present a more favorable option. The specific objective of the research is to create a recommendation guide that assists developers, considering the characteristics of the software to be developed, in deciding between using microservices architecture or monolithic architecture.

The remainder of this paper is structured as follows: Section 2 outlines the research methodology. Section 3 discusses the findings from the analysis of the proposed methodological steps. The conclusion and future work are presented in Section 4.

2 METHODOLOGY

A Multivocal Literature Review (MLR) proposed by Garousi *et al.* (Garousi et al., 2019) was conducted, focusing on the execution of *snowballing* (Wohlin, 2014) - to obtain academic studies on the topic research - as well as in the *Grey Literature Review* guidelines, which consider the knowledge available in non-academic sources, which are relevant to the *software* industry (Figure 1).

Figure 1: Methodology stages.

2.1 Snowballing Process

To understand the academic community's perspective on the increased development time associated with choosing a microservices architecture, the *snowballing* methodology was employed. This systematic study approach starts with an initial selection of articles, using their references and citations for a comprehensive review (Wohlin, 2014). It involves "backward snowballing" (using references) and "forward snowballing" (using citations), with each round of selection considered an "iteration". Iterations continue until no more relevant articles meeting the inclusion and exclusion criteria are found.

The inclusion criteria for the snowballing process and subsequent iterations are: articles discussing the advantages and disadvantages of using microservices; articles in English or Portuguese. The exclusion criteria include: grey literature works, to be covered in the grey literature review; studies not in English or Portuguese; theses and dissertations; books. These results are presented in Section 3.1.

2.2 Grey Literature Review Process

The term "*grey literature*" encompasses a variety of definitions with varying degrees of interconnectedness. The first official definition was introduced at the Third International Conference on Grey Literature in Luxembourg, 1997, conceptualizing it as unpublished primary studies containing information from several sectors like government, academia, business, and the electronic industry, not controlled by commercial publishers. This definition was later expanded in 2004 at the same conference to include academic and industrial reports, as well as dissertations and theses.

In 2010, at the Prague conference, the concept of grey literature was further refined to address intellectual property issues related to internet-based works. Grey literature now includes documents from governmental, academic, business, and industrial levels, in both print and electronic forms, protected by intellectual property rights and of sufficient quality for collection and preservation by libraries or institutions, without being under the control of primarily publishing companies. Notably, this definition excludes blog posts and tweets as grey literature.

Adams *et al.* (Adams et al., 2016) initiated in 2017 a formal evaluation on incorporating grey literature in organizational studies, structuring it into categories reflecting their scientific reliability. The first category (*tier 1*) encompassed sources with stringent source control, such as books, journals, and governmental reports. The second category (*tier 2*) included materials with moderate reliability, like corporate annual reports and media content. The third category (*tier 3*) was designated for sources with lower academic credibility, such as blogs and social media posts. This approach aimed to ensure the scientific integrity and relevance of grey literature in scholarly research.

In the Software Engineering context, Garousi *et al.* (Garousi et al., 2019) tailored the concept by Adams *et al.* (Adams et al., 2016) to consider all non-peer-reviewed materials as grey literature. This adaptation was crucial to encompass the vast array of relevant but academically peripheral content in the field.

The grey literature review, complementing the snowballing method findings, is particularly relevant in Software Engineering due to the industry's rich practice-oriented knowledge. This approach aligns with Garousi *et al.*'s rationale (Garousi et al., 2019), emphasizing the importance of studies that reflect real-world professional experiences, a perspective supported by additional researchers (Kamei, 2020; Soldani, 2020) advocating for more industry-representative studies in Software Engineering.

The grey literature review phase was initiated in parallel with the snowballing approach. To identify pertinent documents, a series of search strings were crafted and applied within Google's search[1] functionality. The specific search terms employed are described as follows: *("comparison" "advantages" "disadvantages" "microservices" "monolith" site*:"intended domain address").

The choice to include the terms advantages and disadvantages was due to the fact that this search could provide results showing the advantages and disadvantages of using monolithic and microservices architectures. Similarly, the inclusion of the terms monolith and microservices was motivated by the search for articles comparing one architecture with the other. The domain filter was chosen because of the need to refine the search, due to the number of results obtained. The six sites selected were chosen because they are recognized by the *software* development community and are listed as follows: *Stack Overflow blog*; *Site Point*; *GFG*; *freeCodeCamp*; *DZone*; *Medium*.

The following inclusion criteria were used to select the papers from the grey literature: text content; texts from the selected *blogs*, due to the fact that there is a large amount of content shared by the community on this type of platform; content that addresses the issue of comparing microservice and monolith architectures; texts in English or Portuguese. The items that formed the criteria for excluding grey literature works are as follows: audio or video content without text transcription; texts not originating from *blogs*; duplicate texts; texts that do not compare the proposed architectures; texts in languages other than English or Portuguese. The results obtained from the review of the grey literature are shown in Section 3.2.

2.3 Survey

Although the previous stages of systematic literature analysis through snowballing, combined with the systematic review of grey literature, can already add value to this research (Runeson and Höst, 2009), we have decided to expand the data source by conducting a survey with experts.

In this stage, the objective was to identify, using an exploratory qualitative approach, the understanding of the use of proposed software architectures from interviews with industry practitioners. Qualitative research denotes a type of research that produces results not resulting from statistical or other quantifiable procedures, but that qualifies the phenomenon under investigation (Corbin and Strauss, 2008). Studies of this nature refer to research about people's lives, experiences, behaviors, emotions, and feelings, as well as organizational functioning, social movements, cultural phenomena, and interactions between people (Corbin and Strauss, 2008). Thus, qualitative research involves the interpretation of the subjects studied, examining things in their natural settings in an attempt to interpret the phenomenon based on the accounts provided by the research participants (Taylor, 2005).

For the field study, the survey method was used, which, based on the theme of this research, employed a questionnaire that was administered to participants during interview sessions. The questionnaire was designed in a way that the questions would result in answers that add value to this work and that they were also engaging, so that people would be more inclined to provide complete and accurate responses if they could see that the study's results would likely be useful to the community. The basis for formulating the questions proposed to the interviewees came from the analysis of the results of the methodologies applied in previous steps of this same research.

Additionally, in order to propose an instrument of value for a representative sample, a pilot test was conducted with the initial interviewees. This allowed for the identification of issues with the questionnaire as well as the follow-up rate procedure. The pilot test was also useful in contributing to the assessment of reliability (Singer et al., 2008).

2.3.1 Interview Process

Semi-structured interviews were conducted following the guidelines associated with Software Engineering (Hove and Anda, 2005) with 7 participants who accepted the invitation. The interviews were conducted, transcribed, and analyzed by the author of this study. For the interviews, questions were formulated based on the results of previous methodological steps. All participants were asked the initial set of questions; however, as is common in semi-structured interviews, new questions emerged during the interviews based on specific responses. This allowed the participants to freely express their perceptions and opinions.

[1] https://www.google.com/

3 RESULTS ANALYSIS

This study employed the snowballing methodology for the systematic collection and analysis of academic data, expanding the sample through citations and references from initial studies to gain a broader understanding of the field (Wohlin, 2014). It also incorporated an analysis of grey literature, including non-commercial sources, for a more comprehensive view (Garousi et al., 2019). Moreover, a survey with software development professionals was conducted to gather their perceptions and opinions, complementing the theoretical evaluation with practical insights. Together, these methods provided an in-depth understanding of the use of software architectures under study.

3.1 Snowballing

For the systematic study using the snowballing technique, the initial selection of articles was extracted from the *Association of Computer Machinery* (ACM) - Digital Library and *IEEE Xplore* databases. To achieve this initial selection, the previously cited search string was employed, with filters being applied according to predefined inclusion criteria. As a result of this process, a total of 41 articles was identified. Of these, 28 were from ACM and the remaining 13 were from the IEEE Xplore database. All abstracts of these articles were carefully read and analyzed.

After analyzing the abstracts, 29 articles from the initial list were excluded. The main reason for the removal of these articles was that their themes were not aligned with the main focus of this research. Additionally, some articles were excluded based on the previously established inclusion and exclusion criteria. This resulted in a total of 12 articles that were selected for the first iteration of the snowballing review process.

The execution of the initial snowballing search enabled the acquisition of 12 articles that were analyzed to extract the advantages and disadvantages present in monolithic and microservices architectures. This identification and categorization work was essential to deepen the understanding of these two architectures and provide a solid foundation for the main objective of this research.

One of the main points of interest of this study is to understand how the adoption of microservices architecture can lead to a longer software development time.

It is important to emphasize that this research does not intend to offer a definitive or exhaustive view, but rather to provide a starting point for a more in-depth discussion on the topic. Each disadvantage presented below represents a characteristic that should be carefully considered by development teams and decision-makers when evaluating the adoption of microservices architecture in their projects.

After the initial selection of articles and analysis of their contents, the iteration process began, involving the use of backward and forward snowballing techniques. This approach allowed for the inclusion of 14 additional articles to the initial set, resulting in a significant increase in information sources for the research (see Table 1).

Table 1: Final selection of articles by Snowballing.

ID	Reference
S1	(Kyryk et al., 2022)
S2	(Gos and Zabierowski, 2020)
S3	(Vidiečcan and Bobák, 2022)
S4	(Raharjo et al., 2022)
S5	(Gördesli and Varol, 2022)
S6	(Velepucha and Flores, 2023)
S7	(Akbulut and Perros, 2019)
S8	(Afanasev et al., 2017)
S9	(Bian et al., 2022)
S10	(Wei et al., 2021)
S11	(Taibi et al., 2017)
S12	(de Oliveira Rosa et al., 2020)
S13	(Salii et al., 2023)

ID	Reference
S14	(Di Francesco et al., 2018)
S15	(Ponce et al., 2019)
S16	(Benavente et al., 2022)
S17	(Christian et al., 2023)
S18	(Blinowski et al., 2022)
S19	(Karabey Aksakalli et al., 2021)
S20	(Kumar et al., 2021)
S21	(Liu et al., 2020)
S22	(Jamshidi et al., 2018)
S23	(Kurpad et al., 2023)
S24	(Butzin et al., 2016)
S25	(Hassan and Bahsoon, 2016)
S26	(Balalaie et al., 2016)

The added articles underwent detailed analysis, during which the main points related to the disadvantages of using microservices were extracted. Notably, already in the second iteration of the snowballing process, no new disadvantages beyond those previously listed were identified, suggesting a saturation of information on the topics of interest.

Given that subsequent iterations did not produce new information, it was decided to conclude the iterations proposed by the snowballing. This decision was based on the assessment that further iterations would not bring significant new data, and that the time and resources invested would be better used in analyzing the data already collected.

The 26 articles selected for the research were read in full to ensure a complete understanding of their content and the context in which they were written.

Regarding the disadvantages associated with the use of microservices architectures, a complete list of all identified disadvantages, as well as the frequency of their occurrences in all analyzed articles, is available in Table 2. This table provides a comprehensive view of the perceptions and experiences reported in the literature on the use of these architectures.

In the analysis of the results obtained through the snowballing process, several challenges associated with the implementation of microservices architecture that can impact software development time

Table 2: Disadvantages associated with the use of microservices.

Disadvantages	Quotes
Operational complexity	20
Initial cost of adoption	12
Data consistency challenges	11
Network overhead	9
Increasing complexity of DevOps	9
More complex monitoring	5
Communication challenges	4
Debugging difficulties	4
Versioning difficulties	4
Challenging security standards	4

were identified. Operational complexity is one of the main issues, as the multiplicity of autonomous services requires intensive management and coordination, increasing the workload.

Other notable challenges include the initial cost of adoption, which involves acquiring specific technical skills and implementing new tools and processes. Additionally, data consistency is a recurring obstacle, as each service manages its own database, leading to data fragmentation and making it difficult to maintain consistency. Network overhead is also a significant challenge, as constant communication between services can result in latency and degraded performance.

Increased complexity in DevOps, communication challenges between services, difficulties in debugging and versioning, as well as challenging security standards are other obstacles that can prolong the software development cycle. Despite the benefits of microservices architecture, such as scalability and flexibility, it is crucial for organizations to be aware of these challenges and be prepared to manage them effectively. The decision to adopt this architecture should therefore be carefully weighed, considering both the potential benefits and inherent challenges.

3.2 Grey Literature

The results obtained from internet blog searches totaled 90 articles. Among these articles, 68 were obtained from Medium blogs, 1 from Site Point, 6 from Geeks For Geeks, 1 from freeCodeCamp, 13 from DZone, and 1 from the Stack Overflow blog.

For the selection of articles according to the inclusion and exclusion criteria, the texts were read and the disadvantages mentioned by them were classified. As some of the blogs contained a large number of results, this classification of the disadvantages of using microservices architecture was useful for generating a knowledge base on these, which showed that the industry has a comprehensive perception of the problems related to the use of this architecture. The final number of selected texts was 27 articles, according to the proposed exclusion criteria. Among these articles, 20 were obtained from the Medium blog, 1 from Site Point, 2 from Geeks For Geeks, 1 from freeCodeCamp, 3 from DZone, and none from the Stack Overflow blog, as seen in Table 3.

Table 3: Results from the grey literature review.

Site	Results	Selected
Medium	68	20
DZone	13	3
Geeks For Geeks	6	2
Site Point	1	1
freeCodeCamp	1	1
Stack Overflow blog	1	0

Similarly to what was found in academia with the use of snowballing, the study of the selected texts by analyzing the grey literature reveals a series of challenges associated with the implementation and use of microservices architecture. These challenges, although varying in degree and complexity, highlight the inherent difficulties of this architecture and the issues that need to be considered when adopting it.

First of all, complexity in implementation and maintenance is cited in 25 of the analyzed texts. This challenge is due to the fact that microservices architecture involves the creation and management of multiple independent services, each with its own lifecycle. This can lead to significant complexity both in the implementation phase and in the maintenance of these services.

Secondly, challenges in managing distributed state are mentioned in 18 of the texts. This problem arises because, in a microservices architecture, the application state is generally distributed among various services. This can make managing the application state complex and prone to errors.

Furthermore, difficulties in deploy automation and configuration are mentioned in 13 of the texts. Automation is essential for efficiently managing the large number of services in a microservices architecture. However, automation can be difficult due to the complexity and diversity of the services.

Additional infrastructure costs, cited in 12 of the texts, are also a concern. As each service in a microservices architecture is usually run in its own environment, this can lead to increased infrastructure costs.

Other issues identified in the texts include increased network latency due to communication between services (11 texts), the need to deal with a variety of technologies (11 texts), difficulties in conducting comprehensive and effective testing (11 texts), security challenges arising from the complexity of the microservices architecture (8 texts), the need for an entry point system to facilitate communication and control between services (3 texts), the lack of adequate training for development teams (2 texts), and challenges in managing and controlling code versioning (2 texts).

The distribution of these issues can be visualized in Table 4. This provides a comprehensive view of the challenges associated with microservices architecture and can help guide future research and practices in this field.

Table 4: Problems found in the grey literature review with the use of microservices architecture.

Listed Problems	Quotes
High complexity in implementation and maintenance	25
Challenges in distributed state management	18
Difficulties in automating *deploy*	13
Additional infrastructure costs	12
Increased network latency due to communication	11
Need to deal with diversity of technologies	11
Difficulties in conducting comprehensive tests	11
Security challenges arising from complexity	8
Need for a point of entry system	3
Lack of adequate training for teams	2
Management and control of code versioning	2

3.3 Survey

In this study, semi-structured interviews were conducted to collect qualitative data on developers' perceptions and experiences with microservices and monolithic architectures. The seven respondents represent a variety of roles, experiences ranging from 3 to 16 years of experience (avg. +8 years), and industry sectors (such as ERP development, travel and others), providing a comprehensive insight into the subject.

The analysis reveals that both types of architecture have their advantages and disadvantages. Microservices architectures, while more complex, offer greater scalability and ease of maintenance, especially in large and complex projects. However, they can result in longer development times, particularly in the initial stages or in less complex projects. On the other hand, monolithic architectures are easier to develop and test due to their simplicity and centralization but can become more challenging to manage in larger and more complex projects.

The respondents agree that the choice of architecture depends on several factors, including the complexity and size of the project, as well as the skills and experiences of the developers. Moreover, the organization and management of the project may be more relevant to the success of the development than the choice of architecture itself. Therefore, the choice of architecture should be based on a careful assessment of the project's needs, organizational context, available resources, and team experience.

3.4 Threats to Validity

Any findings and insights from this study are limited by the nature of the sub-studies of which it is composed. Each study has associated limitations, but some of these are more generally applied.

First, this study could have been submitted to the research bias, where the researcher expects a result and thus interprets findings from this perspective. This study has put in practice techniques which reduce this bias, such as multi-author analysis and discussion of results.

Given that this study has made use of interviews, recordings and transcriptions, there is the limitation towards transcription of interviews and interpretation of responses by the researcher. This is mitigated by the scientific procedures adopted, but cannot be fully eliminated.

It is important to remark that our study involved interviews with only seven experts in the field. While this number may seem modest, the depth and breadth of insights provided by these seasoned specialists significantly enrich our research. Their extensive experience and profound understanding of software architectures lend our findings a depth that might not be achievable through larger, less specialized participant pools. This approach underscores the quality over quantity principle, ensuring our conclusions are both informed and nuanced.

Furthermore, the rapidly evolving field of software architecture means our results might be context-bound or lose relevance over time. Recognizing these limitations is crucial for a balanced understanding of the study's scope and applicability.

4 CONCLUSION

The research conducted in this study provided a comprehensive analysis of microservices and monolithic architectures, considering their advantages and disadvantages and how the choice between them can impact software development time. The results ob-

tained through snowballing methodologies, grey literature, and surveys with industry professionals indicate that while microservices architecture offers significant benefits in terms of scalability, flexibility, and fault isolation, it also presents challenges that can lead to an increase in development time.

The challenges associated with microservices architecture, such as operational complexity, initial adoption cost, data consistency challenges, network overhead, and increased DevOps complexity, require careful management and may demand significant time and resource investment. However, for large and complex projects, microservices may be the better option, offering advantages that outweigh the disadvantages.

This research contributed to the existing literature by providing a detailed view of the implications of architectural choice on software development time. The findings reinforce the need for a thorough assessment of the project's needs and available resources before deciding between microservices and monoliths.

4.1 Future Works

In relation to future works, it is recommended to continue the investigation into microservices and monolithic architectures, with a particular focus on optimizing development time and mitigating identified challenges. Further research could explore strategies and tools that facilitate the transition from monoliths to microservices, as well as practices that promote efficiency in the development and maintenance of microservices-based architectures.

Another area of interest is the development of frameworks that automate aspects of microservices development, reducing operational complexity and speeding up the development cycle. Moreover, detailed case studies in different industry contexts could provide practical data on how companies are addressing the challenges associated with each architecture and which solutions are proving effective.

Finally, future research could focus on the education and training of developers to work with microservices, identifying essential skills and creating educational materials that better prepare professionals for the challenges of this architectural paradigm.

ACKNOWLEDGEMENT

This study was partially supported by the Ministry of Science, Technology, and Innovations from Brazil, with resources from Law No. 8.248, dated October 23, 1991, within the scope of PPI-SOFTEX, coordinated by Softex.

REFERENCES

Adams, R. D., Smart, P., and Huff, A. S. (2016). Shades of grey: Guidelines for working with the grey literature in systematic reviews for management and organizational studies. *International Journal of Management Reviews*, 19:432–454.

Afanasev, M. Y., Fedosov, Y. V., Krylova, A. A., and Shorokhov, S. A. (2017). An application of microservices architecture pattern to create a modular computer numerical control system. In *2017 20th Conference of Open Innovations Association*, pages 10–19.

Akbulut, A. and Perros, H. G. (2019). Performance analysis of microservice design patterns. *IEEE Internet Computing*, 23(6):19–27.

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016). Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Soft.*, 33(3):42–52.

Benavente, V., Yantas, L., Moscol, I., Rodriguez, C., Inquilla, R., and Pomachagua, Y. (2022). Comparative analysis of microservices and monolithic architecture. In *2022 14th Int. Conf. on Computational Intelligence and Communication Networks*, pages 177–184.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84.

Bian, Y., Ma, D., Zou, Q., and Yue, W. (2022). A multi-way access portal website construction scheme. In *2022 5th Int. Conf. on AI and Big Data*, pages 589–592.

Blinowski, G., Ojdowska, A., and Przybyłek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, 10:20357–20374.

Butzin, B., Golatowski, F., and Timmermann, D. (2016). Microservices approach for the internet of things. In *2016 IEEE 21st Int. Conf. on Emerging Technologies and Factory Automation*, pages 1–6.

Christian, J., Steven, Kurniawan, A., and Anggreainy, M. S. (2023). Analyzing microservices and monolithic systems: Key factors in architecture, development, and operations. In *2023 6th Int. Conf. of Computer and Informatics Engineering*, pages 64–69.

Corbin, J. and Strauss, A. (2008). Qualitative research. *Techniques and procedures for developing grounded theory*, 3.

de Oliveira Rosa, T., Daniel, J. a. F. L., Guerra, E. M., and Goldman, A. (2020). A method for architectural tradeoff analysis based on patterns: Evaluating microservices structural attributes. In *Proceedings of the EuroPLoP 2020*, pages 35:1–8. ACM.

Di Francesco, P., Lago, P., and Malavolta, I. (2018). Migrating towards microservice architectures: An industrial survey. In *2018 IEEE Int. Conf. on Software Architecture (ICSA)*, pages 29–2909.

Garousi, V., Felderer, M., and Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. *Info. and Software Technology*, 106:101–121.

Gos, K. and Zabierowski, W. (2020). The comparison of microservice and monolithic architecture. In *2020*

IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), pages 150–153.

Gördesli, M. and Varol, A. (2022). Comparing interservice communications of microservices for e-commerce industry. In *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–4.

Hassan, S. and Bahsoon, R. (2016). Microservices and their design trade-offs: A self-adaptive roadmap. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 813–818.

Hove, S. and Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE METRICS)*, pages 10 pp.–23.

Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., and Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. *IEEE Software*, 35(3):24–35.

Kamei, F. K. (2020). The use of grey literature review as evidence for practitioners. *SIGSOFT Softw. Eng. Notes*, 44(3):23.

Karabey Aksakalli, I., Celik, T., Can, A. B., and Tekinerdogan, B. (2021). Deployment and communication patterns in microservice architectures: A systematic literature review. *Journal of Sys. and Soft.*, 180:111014.

Kholy, M. E. and Fatatry, A. E. (2019). Framework for interaction between databases and microservice architecture. *IT Professional*, 21(5):57–63.

Kumar, P. K., Agarwal, R., Shivaprasad, R., Sitaram, D., and Kalambur, S. (2021). Performance characterization of communication protocols in microservice applications. In *2021 SmartNets*, pages 1–5.

Kurpad, S., BT, S., Vijaykumar, S., Jain, S., and Kalambur, S. (2023). Microarchitectural analysis and characterization of performance overheads in service meshes with kubernetes. In *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–6.

Kyryk, M., Tymchenko, O., Pleskanka, N., and Pleskanka, M. (2022). Methods and process of service migration from monolithic architecture to microservices. In *2022 IEEE 16th Int. Conf. on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, pages 553–558.

Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H., and Li, Z. (2020). Microservices: architecture, container, and challenges. In *2020 IEEE 20th Int. Conf. on Software Quality, Reliability and Security Companion (QRS-C)*, pages 629–635.

Merson, P. and Yoder, J. (2020). Modeling microservices with ddd. In *2020 IEEE Int. Conf. on Software Architecture Companion (ICSA-C)*, pages 7–8.

Ponce, F., Márquez, G., and Astudillo, H. (2019). Migrating from monolithic architecture to microservices: A rapid review. In *2019 38th Int. Conf. of the Chilean Computer Science Society (SCCC)*, pages 1–7.

Raharjo, A. B., Andyartha, P. K., Wijaya, W. H., Purwananto, Y., Purwitasari, D., and Juniarta, N. (2022). Reliability evaluation of microservices and monolithic architectures. In *Int. Conf. on Computer Engineering, Network, and Intelligent Multimedia*, pages 1–7.

Richardson, C. (2016). Pattern: Monolithic architecture (2014). *URL http://microservices. io/patterns/monolithic. html*.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Soft. Eng.*, 14:131–164.

Salii, S., Ajdari, J., and Zenuni, X. (2023). Migrating to a microservice architecture: benefits and challenges. In *2023 46th MIPRO*, pages 1670–1677.

Singer, J., Sim, S. E., and Lethbridge, T. C. (2008). *Software Engineering Data Collection for Field Studies*, pages 9–34. Springer London, London.

Soldani, J. (2020). Grey literature: A safe bridge between academy and industry? *SIGSOFT Softw. Eng. Notes*, 44(3):11–12.

Soldani, J., Tamburri, D. A., and Van Den Heuvel, W.-J. (2018). The pains and gains of microservices: A systematic grey literature review. *Journal of Systems and Software*, 146:215–232.

Stubbs, J., Moreira, W., and Dooley, R. (2015). Distributed systems of microservices using docker and serfnode. In *2015 7th International Workshop on Science Gateways*, pages 34–39.

Taibi, D., Lenarduzzi, V., Pahl, C., and Janes, A. (2017). Microservices in agile software development: A workshop-based study into issues, advantages, and disadvantages. In *Proceedings of the XP2017 Scientific Workshops*, XP '17, New York, NY, USA. Association for Computing Machinery.

Taylor, G. R. (2005). *Integrating quantitative and qualitative methods in research*. University press of America.

Velepucha, V. and Flores, P. (2023). A survey on microservices architecture: Principles, patterns and migration challenges. *IEEE Access*, 11:88339–88358.

Vidiečcan, M. and Bobák, M. (2022). Container-based video streaming service. In *2022 IEEE 22nd International Symposium on Computational Intelligence and Informatics and 8th IEEE International Conference on Recent Achievements in Mechatronics, Automation, Computer Science and Robotics (CINTI-MACRo)*, pages 000191–000196.

Viennot, N., Lécuyer, M., Bell, J., Geambasu, R., and Nieh, J. (2015). Synapse: A microservices architecture for heterogeneous-database web applications. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 21:1–16. ACM.

Wei, Y., Yu, Y., Pan, M., and Zhang, T. (2021). A feature table approach to decomposing monolithic applications into microservices. In *Proc. of the 12th Asia-Pacific Symposium on Internetware*, Internetware '20, page 21–30, Singapore, Singapore.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *EASE*, pages 38:1–10, London, UK.

Yarygina, T. and Bagge, A. H. (2018). Overcoming security challenges in microservice architectures. In *2018 IEEE SOSE*, pages 11–20.