Date available online: 13-02-2025
Vol. 2, Issue 2 (June - December 2024)

A Design-Oriented Classification of Microservice Smells

Junaid Aziz[1], and Ghulam Rasool[1]

[1]Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan

Corresponding author: Junaid Aziz (e-mail: mjunaidaziz@gmail.com).

***Abstract*— Context:** Introduction of bad smells can generate negative consequences on the quality of microservices. It is essential to gather state-of-the-art knowledge on these smells and understand the challenges they present. This will benefit researchers and practitioners in mitigating the consequences of smells in microservice-based systems. **Objective:** The main goal of this study is to present a comprehensive catalogue of microservice smells. **Method:** To document the advancements and best practices in the field of microservice smells., we performed a multivocal literature review study incorporating both academic and grey literature sources. We systematically analyzed 34 studies published from the beginning of 2014 until the end of 2023 by following standard guidelines. **Results:** 38 bad smells in microservices are identified and cataloged in 10 different types. **Conclusion:** Research gaps and open challenges are highlighted in this study. This will give directions to other researchers and practitioners towards addressing challenges posed by smells in microservices.

Index Terms—Microservice architecture; Bad smells; Anti-patterns;

I. INTRODUCTION

Microservice architecture (MSA) is powerful and popular architecture style for polyglot applications built with the composition of small services, called “microservices”. In this architecture, the core concept of high cohesion and loose coupling is applied by making these services completely independent in development and deployment. Besides, every service is required to run its processes on its own and communicate with other services via lightweight mechanisms. Service splitting, testing, integration of services, logging, and monitoring are some of the technical challenges that need to be addressed with the help of design patterns while adopting MSA [1].

Antipatterns or bad smells are symptoms that may indicate a deeper quality problem in the system design or code [2]. These smells can be found at various levels of abstraction, such as architecture, design, and code [3]. Code smells are flaws in the design of software that makes it difficult to comprehend and maintain compatibility, resulting in less resilient and compromised system development [4]. Design smells can influence a set of classes in a design framework and highlight violations of design principles such as the principle of circular dependencies [5]. The architecture smell is a higher-level system design issue. This is a sort of technical debt that can have a negative impact on the overall maintainability of a software system. Compared to refactoring code and design smells, refactoring architecture smells takes more time and effort [6].

To counter smells in applications, the software engineering community has explored various ways including proposing catalogs of smells for MSA. However, the scattered information about such resources poses a challenge to both researchers and practitioners while developing appropriate methods, techniques, and tools concerning microservice smells or antipatterns. Hence, analyzing and synthesizing information from academic and grey literature might help the software development community in comprehending the current state of knowledge on microservice smells. Recently, a tertiary study [48] is performed to consolidate a catalog of microservice smells extracted only from academic literature. Microservice smells reported in grey literature are missing in their study. This multivocal literature review (MLR) fill this gap by extending their work. The objective of this MLR is to consolidate both academic and industrial knowledge in order to capture the state of art and practice on microservice smells by including only those microservice smells which have been frequently discussed among researchers and practitioners. The following are the major novel contributions of this study:

- Highlight most frequently studied microservice smells in the literature
- Present a catalog of microservice smells based on their types

The remainder of this paper is arranged as follows. In related work we discuss the existing studies performed on microservice smells and highlight their shortcomings. Research methodology followed in the study is described in survey methodology section. Results of the study along with discussion are presented in results and discussion section. Conclusions section outline the outcome of this study and potential future directions.

II. RELATED WORK

There have been few attempts aiming at reviewing the state-of-the-art and current practices on microservice smells. An overview of these studies is illustrated here.

Neri et al. [7]conducted a systematic review of academic and grey literature to identify the most well-known architectural smells in microservices. They also proposed a taxonomy of these smells by organizing them based on four design principles such as independent deployability, horizontal scalability, isolation of

failure, and decentralization. Taxonomy proposed in their study is missing a lot of microservice smells that have discovered recently such as no service template, local logging, influential service, etc.

Soldani et al. [8]discovered through a grey literature review that microservices-based applications have to face design, development, and operational challenges. They discovered that in such applications, business logic is heavily dispersed among a large number of microservices that are all evolving independently. They also found that there is a lack of methodology to quantify and minimize bad design decisions in such types of applications. Their study was performed with a focus on just technical challenges faced by microservices.

Bogner et al. [3] conducted a literature review of microservices bad smells and antipatterns. Their survey was not only conducted on a limited number of digital libraries but also included antipatterns and bad smells highlighted by researchers only. Studies from grey literature were not included in their search process. Moreover, they did not report or classify the harmfulness of antipatterns.

Tighilt et al. [9]conducted a literature review of published articles and analyzed different open-sourced projects. They presented a catalog of microservice antipatterns (or smells) along with their implementation and refactoring solutions to remove them. However, the viability of refactoring solutions proposed in their study was not empirically validated. Besides, they did not mention smells related to testing and organization.

Carrasco et al. [21] found 9 bad smells related to MSA and its migration by digesting different sources from the academia and grey literature. Their study revealed some common best practices as potential solutions for the architecture and migration bad smells but ignoring security and monitoring smells. Additionally, their search for academic and grey literature sources was performed using search engines only. This gives rise to doubts about the credibility and completeness of results.

The authors conducted a tertiary study aiming not only to compile a comprehensive catalog of overlapping microservice smells but also to categorize them [48]. Their catalog is based on material drawn from academic literature, overlooking the smells reported in grey literature. This restriction may impact the comprehensiveness of the catalog, as grey literature often encompasses valuable insights and practical experiences that contribute to an in depth understanding of microservice smells.

Our work is different from these studies as we are aiming here to build a unified catalog of all microservice smells reported in academic and grey literature both. This study presents these smells in a classified form that has been generated based on the nature of each smell. This will help in achieving standardization vis-à-vis microservices as suggested in [45].

III. RESEARCH METHODOLOGY

Multivocal literature review (MLR) is a type of literature review that incorporates both academic and grey literature sources. Academic literature includes material that is peer-reviewed such as papers published in journals and conferences. Grey literature consists of material that is publicly available in different forms such as blogs, videos, white papers, books, etc. Unlike academic literature, grey literature usually does not undergo a rigorous peer-review process. MLRs are useful for both researchers and practitioners because they summarize the current state of the art from academic and grey literature on a specific topic.

We have followed the guidelines proposed by Garousi et al. [10] to perform this MLR, which are based on systematic literature review (SLR) guidelines suggested by Kitchenham et al. [11]. As per the guidelines, the MLR review process involves three major stages: planning the review, conducting the review, and reporting the review.

A. PLANNING THE MLR

The objective of this study is to capture the state of the art and practices in cataloguing microservice smells. During initial observation it is found that due to their strong interest on microservices both researchers and practitioners have contributed a lot through academic and grey literatures. Hence, instead of conducting either systematic literature review or mapping study, a comprehensive MLR is conducted to capture the state of the art and practices in line with the objectives of this study. To achieve this, we classify peer-reviewed papers (i.e., journal and conference articles) as academic literature and other studies (i.e., blog posts, industrial whitepapers, articles, videos, and books) as grey literature. Moreover, we have formulated the following research questions for this study: -

RQ: How much attention different types of microservice smells have received from academia and industry?

Rationale – By consolidating the list of reported smells from both academic research and industry sources, our goal is to categorize the smells found in microservices-based applications. This approach will facilitate the creation of a unified catalog of these smells which is currently lacking.

B. CONDUCTING THE MLR

We employed IEEE Xplore, ACM, Springer, ScienceDirect, DBLP and Scopus to search papers in the academic literature which are widely used databases and digital libraries to extract computer science and software engineering publications [25]. To look for grey literature (e.g., books, blog posts, videos, whitepapers), we used Google, Bing and DuckDuckGo search engines. The reason for using these search engines is that they remain consistent over a period of time and a considerable difference in the results produced by them is witnessed with Google standing apart [107]. We looked for published studies between the beginning of 2014 (when Lewis et al. [1] introduced microservices) and the end of 2023. The search string "smell OR antipattern OR anti-pattern OR debt OR anomal) AND (microservice OR micro-service" was structured according to the criteria suggested by Petersen et al. [13]. The search string includes keywords from each aspect of our study problem. We ran the search string on academic and grey literatures independently. Final selection of studies from both academic and grey literatures were merged later. The stages of our search and selection process for academic literature and grey literatures can be seen in Figure 1. Authors performed

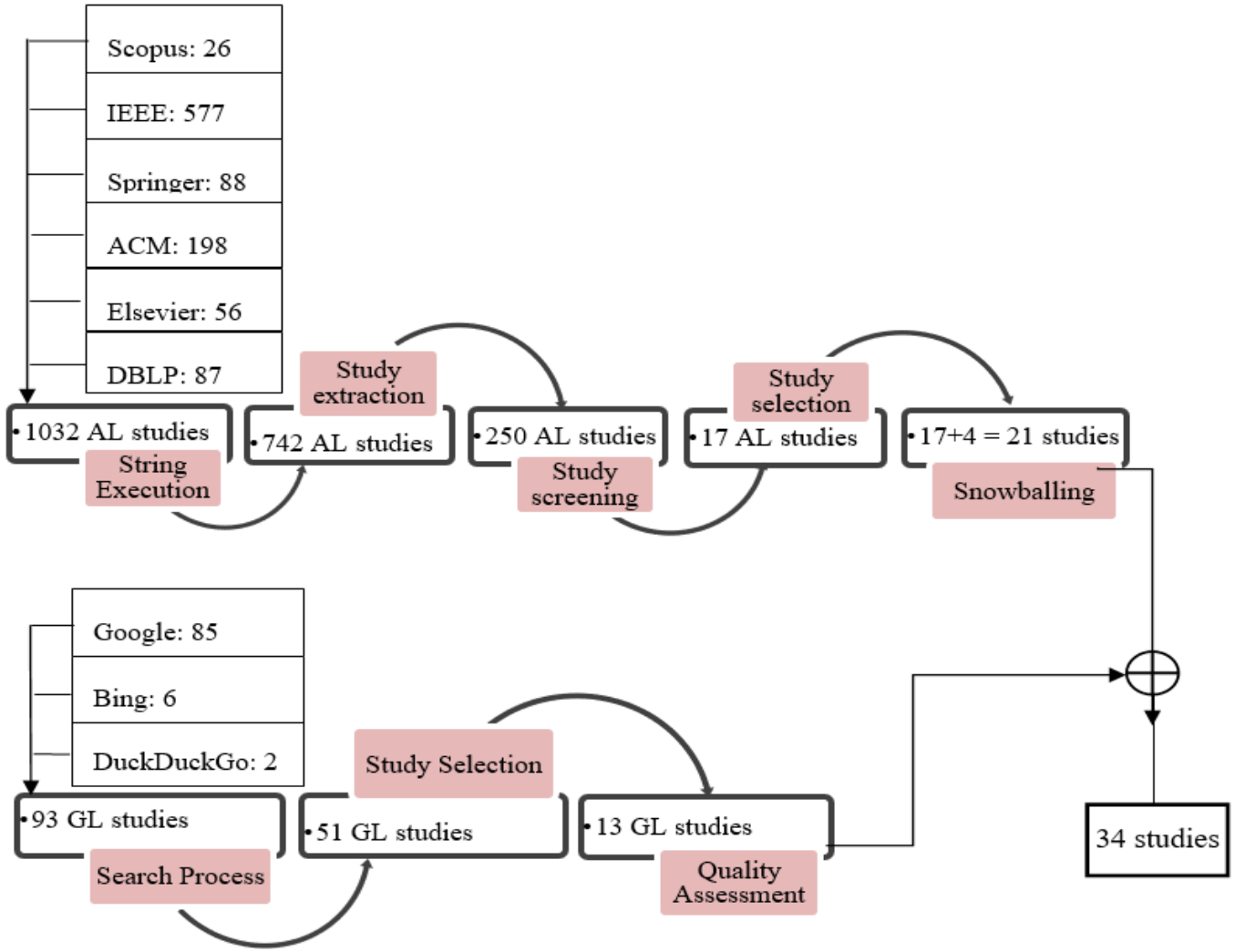

FIGURE.1. MLR search process (AL=academic literature, GL=grey literature)

these steps iteratively and the final selection of studies was made with consensus whereas conflicts were resolved through the mediation of another researcher.

Step1 — String execution: The search string was applied to the title, abstract, and keywords of studies in all electronic databases (see Table 3).

Step2 — Study extraction: We marked the study as "relevant" in our datasheet if its title or keywords were matching the search terms of this study to keep it for future reading. Otherwise, it was ruled out. Duplicates were also removed from the selected studies.

Step3 — Study screening: Each study was thoroughly examined for additional processing by reading the abstracts and conclusions.

Step4 — Study selection: We included a study for this MLR if it met all of the inclusion criteria and none of the exclusion criteria after reviewing the complete text of the study (see Table 1). This yielded us 17 studies from academic literature and 13 studies from the grey literature.

Step5 — Snowballing: For further identification of relevant studies, we examined the references of selected studies using Wohlin [27] forward and backward snowballing techniques. This helped us to identify 4 additional primary studies from academic literature which were not found in the initial search. The final set of articles contains 21 academic studies (see Table A.2 in Appendix A), and 13 studies from the grey literature (see Table A.1 in Appendix A).

TABLE 1
PRIMARY STUDIES SELECTION CRITERIA

Inclusion	Exclusion
- Journal/Conference Articles, blog posts, whitepapers, industry articles, videos, and books written in English - The search terms or synonyms are present in title, keywords or comments - In abstract or summary authors are specifically addressing microservices Smells or related terms - The contribution of the academic/grey literature studies need to be clearly described in terms of techniques applied, model evaluation with least 1 case study, issues faced concerning smells, lessons learned, and limitations	- A study that is found unsuitable by assessing either the title or abstract or summary - A study that has no full text available - Sufficient focus on smells (or related terms like technical debt) and its detection technique is not provided - Studies covering topics such as benefits of Microservice Architecture, Comparing SOA with Microservices, etc. - Studies that are written by practitioners having no known experience in microservices - Studies based on assumptions or simulation

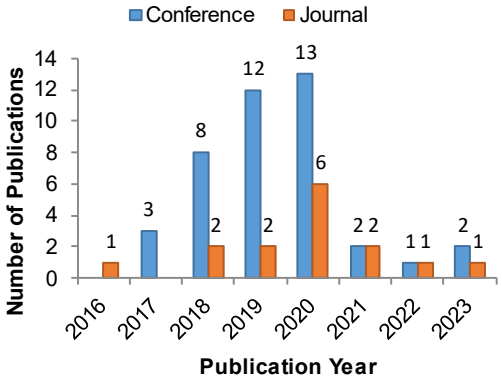

FIGURE 2. Academic literature publication types

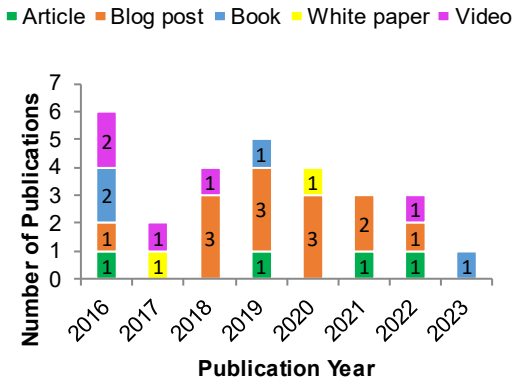

FIGURE 3. Grey literature publication types

IV. RESULTS AND DISCUSSION

This section highlights and discusses the findings obtained after evaluating and synthesizing data from the selected studies that relate to each of the research questions addressed in this study.

A. RQ: How much attention different types of microservice smells have received from academic and industry? RQ: How much attention different types of microservice smells have received from academic and industry?

In the studied time interval, it has been observed that microservice smells started attracting the attention of both communities in 2016 with grey literature studies taking the lead over academic studies. In academic literature from 2017 onwards, conference publications have mostly prevailed over journal articles (see Figure2). Besides, the number of publications in conferences has grown faster as compared to journals which have grown steadily. Sources in grey literature have varied over time (see Figure3). However, most of the contributions on the topic came from professional community blogs.

The following design types were used to classify the selected studies from academic and grey literature: -

- Case study: An industrial problem is chosen for evaluation
- Empirical study: Results are made either by conducting interviews or evaluating more than one real-time application
- Experimental: Evaluation is done through a prototype on small scale problems
- Personal experience: Experience is gained by following the complete lifecycle of an industrial problem
- Solution proposal: Solution is proposed without verifying or evaluation
- Tool: A tool is developed and released for further evaluations

The distribution of studies by design types listed above throughout the studied period in academic and grey literature is shown in Figure 4. It is witnessed that experience gained through solving industrial problems was shared by both communities on yearly basis. Moreover, in academic literature, 16 primary studies were found to have conducted empirical studies. The majority of the information about microservice smells extracted by interviewing microservice developers and practitioners on different forums. As a result of this, few smell detection tools were introduced but detecting a limited number of microservice smells only. Additionally, we did not find any experimental study in grey literature whereas, in academic literature, 13 studies werefound to have performed experimental evaluations through small-scale prototypes. One possible outcome of this trend suggests that academia needs access to more industrial-based microservice systems to improve their proposed techniques and tools for the detection of smells.

We have explored microservice smells with a focus on different areas. These explorations have resulted in various types of smells which have been classified in this study based on the nature of each smell. It is also pertinent to distinguish between severe and non-severe smells. Few studies have attempted to identify smells having negative impact on microservices empirically. In (A15), authors analyzed the documentation of a real life microservice-based project and conducted interviews. Based on the results, they list down smells that are found to have negatively impacted the system without ranking them. In (A6), after conducting survey of experience microservice developers, authors assigned harmfulness score to each microservice smell on a 10-point Likert scale where 0 being not harmful and 10 being extremely harmful. We have used this information and provided severity level of microservice smells in terms of Low (≥ 0), Medium (≥ 4) and High (≥ 6) here in this study. Table 2 presents a catalog of all types of microservice smells identified by this study with corresponding references and their severity levels. By doing so, we intend to remove the ambiguity of similar smells that have been reported under different names causing confusion among researchers and practitioners. Severity column of smells having no such information found by this study is left blank in Table 2. This gives a potential direction to other researchers to empirically evaluate their level of impacts on microservices. This study also found that smells in microservice-based applications may occur not only in various stages of software development but also at the organization level if proper policies are not adopted. This will also lead to an unfruitful migration, especially from monolith to microservices. Therefore, suitable practices and techniques should be adopted to avoid smells at all stages of

application development. Moreover, some disparity about the importance of these types of smells is found in academic and grey literature. For instance, test, data, migration, cloud, monitoring, and security type of smells were discussed more in academic literature whereas the contribution of grey literature was found to be mostly in architecture, design, and organization types of smells. This trend suggests that practitioners might have not witnessed those smells being discussed widely in academic literature or current microservice smell detection tools are not up to the mark. In the future, this information gap may narrow down once appropriate microservice smells detection tools become available and fully operational.

TABLE 2MICROSERVICE SMELLS

Type	Smell	Description	Study ID	Severity
Architecture	Shared Persistence also known as. Data Ownership	Different microservices access the same relational database.	A5,A17, A6, A18,G2, G3,G4,G 9,G10,G7 ,,G10	High
	API Versioning also known as. Static Contract	APIs are not semantically versioned.	A6,A5,A 6,A12,A1 7, A19,A20, G2,G9,G 13,G7,G9	High
	Wrong Cuts also known as. Developer Without a Cause	Occurs when microservices are not split by features	A15,A16, A5,A8,A 2,A6, G2, G12, G7	High
	NO-API Gateway also known as.Service Fan Out	Microservices communicate directly with each other.	A1,A6,A 15,A45, G4,G10, G12, G7, G9	Mediu m
	Cyclic Dependency	A cyclic chain of calls between microservices exists	A3,A5,A 7,A9,A6, A14, G7, G11	High
	Inappropriate Service Intimacy	The microservice keeps on connecting to private data from other services	A5,A6, G12, G7	Mediu m
	Unstable Dependency	A subsystem that depends on other less stable subsystems	A9, G12,G13	-
	ESB Usage	The microservices communicate via an enterprise service bus	A6, G7	High
	Timeout also known as..Are We There Yet	The service consumer is unable to connect to the microservice.	G2,G13	-
	Give It a Rest	Thinking of REST as the only communication platform and ignoring the power of messaging	G2,G3	-
	Queue Explosion	Microservices interact with multiple message queues to get asynchronous guaranteed processing	G8	-
	Hub-Like Dependency	Arises when an abstraction has dependencies with a large number of other abstractions	A9	-
Design	Microservice Greedy also known as. More The Merrier	Teams tend to create new microservices for each feature, even when they are not needed	A1,A5,A 6,A7,A8, A2,A6,A 13, A18,G1, G2, G3,G7,G 10	Low
	Distributed Monolith also known as Sloth	A Service becomes a standalone monolith itself	A7,G5,G 5,G10, G6,G13	-
	Anemic Model	Domain objects contain little or no business logic	A8,G6	-
	Feature Concentration	Occurs when an architectural entity implements different functionalities in a single design construct.	A9	-
Code	Shared Libraries also known as. I Was Taught to Share	Shared libraries between different microservices are used	A3,A6, A18,A19, G2, G7	Mediu m
	Hard-Coded Endpoints also known as Hardcoded IPs and Ports	Hardcoded IP addresses and ports of the services between connected microservices exist	A6, G13,G7	High
	No Service Template	A template that can be used by developers for developing new service	A1,A3,A 5,G7	-
	Local Logging	Logs are stored locally in each microservice, instead of using a distributed logging system	G7	-
Test	Oracle problem also known as. Pride	Output of test results are difficult to verify given an input to the system	A4, G5	-
	Test Endpoints	Team implements additional service endpoints for testing purpose	A8,G10	-
Data	System Referential Integrity	Recovery of the system referential integrity in case of a disaster crash	A4	-
Migration	Complex Legacy	Team finds legacy code buggy or complex so build a new one	A6,A14, A21	-
	Data-driven Migration	It occurs when you migrate both the service functionality and the corresponding data together when creating microservices	G2	-
Organizati on	Too Many Standards also known as Lust/Gluttony	Different development languages, protocols, frameworks are used.	A5,A6,A 5,A6,A12 , A18,A21, G1,G3,G	Mediu m

			4,G5,G6, G7	
	Lack of guidance also known as Magic Pixie Dust	No guided material on how to migrate monolithic to microservice	A6,A13, G1,G2,G 9	-
	Small Team Size also known as. Greed	Team of developers may be assigned to work on more than one microservice due to a shortage of skilled people	A6,A15, G5,G7,G 12	-
	Red Flag also known as. Wrath	The company still work without changing their processes and policies. No CI/CD tools are introduced to developers	G1,G5,G 7,G9	-
	Human Evolvability also known as Scattershot	Teams are often out of sync with respect to the complete picture of a system	A8,A19, G1,G7, G12	-
Cloud	Cold-start	A newly created container incurs a start-up latency due to runtime initialization	A16	-
	Critical Component	A service violates service level objectives (SLO) of the associated request under power capping	A3	-
Monitoring	Trace anomaly	Anomaly propagation from massive monitoring metrics, and to pinpoint the root cause of the failure.	A17,A4, A15,A16	-
	Lack of monitoring	No mechanism to monitor the status of microservices	A6, G7	-
	Influential service also known as Mega-Service	A service becomes critical and may become the cause of system failure/affect cloud power management	A7,G6,G 7	-
	Lack of evaluation methods	Lack of metrics and evaluation methods to check the performance	A7	-
Security	Confused deputy attacks	Trust between microservices is compromised	A2	-
	Powerful tokens	One security token is generated for all the services	A2	-

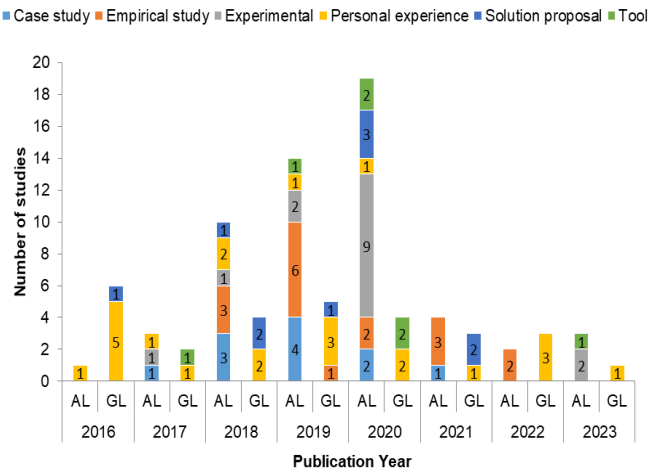

FIGURE 4. Study designs of academic (AL) and grey (GL) literature

B. Open challenges and future research directions

Research in the field of microservice smells is still young and evolving. This study has identified the following open challenges for researchers and practitioners:

1) Need polyglot tools for smells detection and refactoring. The majority of microservice smells detection tools only work with java-based applications. Similarly, in the case of refactoring, there is no difference. This requires attention as MSA provides the flexibility of building applications using multiple programming languages. Researchers and practitioners need to look for avenues where tools can be built for detecting and refactoring diverse types of microservice smells covering different programming languages.

2) Lack of standard benchmark systems for smells detection tools. Only a handful of tools are available to detect a limited number of smells. Moreover, these tools have been validated on different and small toy problems except MSANose [A47] and MARS [A54] which have been tested on two mid-sized benchmark systems. It is also challenging to detect the disparity in the results of these tools due to the availability of a limited number of benchmark systems. Hence, it becomes quite difficult for developers to choose the right tool. There is also a need to devise an evaluation matrix of current and future smells detection tools to help developers in making the correct choice.

3) Need of industrial case studies for finding severe microservice smells. A little disparity about the importance of microservice smells found in academic and grey literature suggests the need for access to more industrial-based microservice systems. This may help researchers in addressing only those smells which are found vulnerable by practitioners. Lotz et al. [98] have performed such an experiment through a case study based on an embedded system and checked the applicability of microservice smells reported in the literature. More similar case studies, covering different domains are needed to scrutinize the currently reported list of microservice smells.

V. CONCLUSION

This study provides a brief and comprehensive overview of up-to-date information about microservice smells. We searched extensively in the academic and grey literature for relevant studies published between 2014 and 2023. This helped us identifying a wide range of smells (38 at the time of writing) and cataloging them into 10 different types as per the nature of each smell. We also found that the community has so far discussed architecture, design and organization smells mostly; with little focus on other types of smells. Also, currently available tools can only detect certain types of smells. This implicates that a complete tool capable of detecting diverse smells in microservices covering multiple programming languages is still lacking.

CONTRIBUTION OF AUTHORS

Junaid Aziz: Conception, Methodology, Investigation and analysis, Writing original draft & editing, Visualization. Ghulam Rasool: Conception, Supervision.

REFERENCES

[1] Microservice API Patterns. Retrieved Mar 1, 2024, from https://microservice-api-patterns.org/

[2] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D., & Gamma, E. (1999). Refactoring: Improving the Design of Existing Code (1st ed.). Addison-Wesley Professional.

[3] Bogner, J., Boceck, T., Popp, M., Tschechlov, D., Wagner, S., & Zimmermann, A. (2019). Towards a Collaborative Repository for the Documentation of Service-Based Antipatterns and Bad Smells. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C).

[4] Yamashita, A., & Moonen, L. (2013). Do developers care about code smells? An exploratory survey. 2013 20th Working Conference on Reverse Engineering (WCRE).

[5] Suryanarayana, G., Samarthyam, G., & Sharma, T. (2014). Refactoring for Software Design Smells: Managing Technical Debt (1st ed.). Morgan Kaufmann.

[6] Fontana, F. A., Pigazzini, I., Roveda, R., & Zanoni, M. (2016). Automatic Detection of Instability Architectural Smells. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME).

[7] Neri, D., Soldani, J., Zimmermann, O., & Brogi, A. (2019). Design principles, architectural smells and refactorings for microservices: a multivocal review. SICS Software-Intensive Cyber-Physical Systems, 1-13.

[8] Soldani, J., Tamburri, D. A., & 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.

[9] Tighilt, R., Abdellatif, M., Moha, N., Mili, H., Boussaidi, G. E., Privat, J., & Guéhéneuc, Y. G. (2020, July). On the Study of Microservices Antipatterns: a Catalog Proposal. In Proceedings of the European Conference on Pattern Languages of Programs 2020 (pp. 1-13).

[10] Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, 106, 101-121.

[11] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software engineering," in "EBSE Technical Report," 2007, vol. EBSE- 2007-01

[12] Cavacini, A., 2015. What is the best database for computer science journal articles?.Scientometrics 102 (3), 2059–2071

[13] Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. In: Proceedings of the 12th international conference on evaluation and assessment in software engineering (EASE'08). BCS Learning & Development Ltd, pp 68–77

[14] Lenarduzzi, V., Lomio, F., Saarimäki, N., & Taibi, D. (2020). Does migrating a monolithic system to microservices decrease the technical debt? Journal of Systems and Software, 169, 110710.

[15] Antonio, N., Vitor, J., Khaled, M., & Ali, A. (2018, October). Fine-Grained Access Control for Microservices. In The 11th International Symposium on Foundations & Practice of Security (Vol. 11358). Springer.

[16] de Toledo, S. S., Martini, A., & Sjøberg, D. I. K. (2020). Improving Agility by Managing Shared Libraries in Microservices. Agile Processes in Software Engineering and Extreme Programming – Workshops, 195–202.

[17] Luo, G., Zheng, X., Liu, H., Xu, R., Nagumothu, D., Janapareddi, R., ... & Liu, X. (2019, December). Verification of microservices using metamorphic testing. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 138-152). Springer, Cham.

[18] Fritzsch, J., Bogner, J., Wagner, S., & Zimmermann, A. (2019, September). Microservices migration in industry: intentions, strategies, and challenges. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 481-490). IEEE.

[19] Zhang, H., Li, S., Jia, Z., Zhong, C., & Zhang, C. (2019, March). Microservice architecture in reality: An industrial inquiry. In 2019 IEEE international conference on software architecture (ICSA) (pp. 51-60). IEEE.

[20] Bogner, J., Fritzsch, J., Wagner, S., & Zimmermann, A. (2019, March). Microservices in industry: insights into technologies, characteristics, and software quality. In 2019 IEEE international conference on software architecture companion (ICSA-C) (pp. 187-195). IEEE.

[21] Bogner, J., Fritzsch, J., Wagner, S., & Zimmermann, A. (2019, September). Assuring the evolvability of microservices: insights into industry practices and challenges. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 546-556). IEEE.

[22] Gouigoux, J. P., & Tamzalit, D. (2017, April). From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. In 2017 IEEE International Conference on Software Architecture Workshops (ICSAW) (pp. 62-65). IEEE.

[23] Hou, X., Li, C., Liu, J., Zhang, L., Hu, Y., & Guo, M. (2020, November). ANT-man: towards agile power management in the microservice era. In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-14). IEEE.

[24] Taibi, D., & Lenarduzzi, V. (2018). On the definition of microservice bad smells. IEEE software, 35(3), 56-62.

[25] Chen, L. (2018, April). Microservices: architecting for continuous delivery and DevOps. In 2018 IEEE International conference on software architecture (ICSA) (pp. 39-397). IEEE.

[26] Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., Larsen, S. T., & Dustdar, S. (2018). Microservices: Migration of a mission critical system. IEEE Transactions on Services Computing.

[27] Abidi, M., Khomh, F., & Guéhéneuc, Y. G. (2019, July). Anti-patterns for multi-language systems. In Proceedings of the 24th European Conference on Pattern Languages of Programs (pp. 1-14).

[28] de Toledo, S. S., Martini, A., Przybyszewska, A., & Sjøberg, D. I. (2019, May). Architectural technical debt in microservices: a case study in a large company. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt) (pp. 78-87). IEEE.

[29] Gunasekaran, J. R., Thinakaran, P., Nachiappan, N. C., Kandemir, M. T., & Das, C. R. (2020, December). Fifer: Tackling resource underutilization in the serverless era. In Proceedings of the 21st International Middleware Conference (pp. 280-295).

[30] Sundelin, A., Gonzalez-Huerta, J., & Wnuk, K. (2020, June). The hidden cost of backward compatibility: when deprecation turns into technical debt-an experience report. In Proceedings of the 3rd International Conference on Technical Debt (pp. 67-76).

[31] Richardson, C. (2019). Microservices adoption antipatterns [Blog]. Retrieved 2020, from https://microservices.io/microservices/antipatterns/-/the/series/2019/06/18/microservices-adoption-antipatterns.html.

[32] Richards, M. (2016). Microservices AntiPatterns and Pitfalls. o'reilly.

[33] Gupta, D., & Palvankar, M. (2020). Pitfalls & Challenges Faced During a Microservices Architecture Implementation. Retrieved from https://www.cognizant.com/whitepapers/pitfalls-and-

challenges-faced-during-a-microservices-architecture-implementation-codex5066.pdf
[34] Abbott, M. (2019). MICROSERVICE ANTI-PATTERN: THE SERVICE MESH [Blog]. Retrieved 2020, from https://akfpartners.com/growth-blog/microservice-anti-pattern-service-mesh.
[35] Bryant, D. (2016). The Seven Deadly Sins of Microservices (Redux). OpenCredo. Retrieved 2020, from https://opencredo.com/blogs/the-seven-deadly-sins-of-microservices-redux/
[36] Tilkov, S. (2018). Microservice Patterns & Antipatterns [Video]. Retrieved 2020, from https://www.youtube.com/watch?v=RsyOkifmamI.
[37] Pitman, C. (2018). Microservice Antipatterns: The Queue Explosion. Retrieved 2021, from http://cpitman.github.io/microservices/2018/03/25/microservice-antipattern-queue-explosion.html#.YIfX9R3ivIU
[38] Alagarasan, V. (2016). Microservices Antipatterns [Video]. Retrieved 2020, from https://www.youtube.com/watch?v=uTGIrzzmcv8
[39] de Toledo, S. S., Martini, A., & Sjøberg, D. I. (2021). Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. Journal of Systems and Software, 177, 110968.
[40] Wang, Y., Kadiyala, H., & Rubin, J. (2021). Promises and challenges of microservices: an exploratory study. Empirical Software Engineering, 26(4), 1-44.
[41] Ramírez, F., Mera-Gómez, C., Bahsoon, R., & Zhang, Y. (2021, June). An Empirical Study on Microservice Software Development. In 2021 IEEE/ACM Joint 9th International Workshop on Software Engineering for Systems-of-Systems and 15th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (SESoS/WDES) (pp. 16-23). IEEE.
[42] Alves, J. (2021), Disasters I've seen in a microservices world, Retrieved 2021, from https://world.hey.com/joaoqalves/disasters-i-ve-seen-in-a-microservices-world-a9137a51,
[43] Kanjilal, J. (2021), Managing Application Dependencies in Distributed Architectures, Retrieved 2021, from https://www.developer.com/design/managing-application-dependencies/
[44] Deeeet (2021), How We Reorganize Microservices Platform Team, Retrieved 2021, from https://engineering.mercari.com/en/blog/entry/20210908-2020-07-16-083548/
[45] Abdelfattah, A. S., & Cerny, T. (2023). Roadmap to Reasoning in Microservice Systems: A Rapid Review. Applied Sciences, 13(3), 1838.
[46] D. Taibi, B. Kehoe and D. Poccia, "Serverless: From Bad Practices to Good Solutions," 2022 IEEE International Conference on Service-Oriented System Engineering (SOSE), Newark, CA, USA, 2022, pp. 85-92.
[47] Cummins, H. (2022, March 17). Seven ways to fail at microservices. Retrieved May 1, 2023, from https://www.infoq.com/presentations/7-microservices-anti-patterns/
[48] Cerny, T., Abdelfattah, A. S., Al Maruf, A., Janes, A., & Taibi, D. (2023). Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. Journal of Systems and Software, 206, 111829.
[49] Taibi, D., Lenarduzzi, V., & Pahl, C. (2020). Microservices anti-patterns: A taxonomy. In Microservices (pp. 111-128). Springer, Cham..

APPENDIX A
STUDIES SELECTED FOR THIS MLR

TABLE A.1
SELECTED STUDIES IN GREY LITERATURE

ID	Title	Reference
G1	Microservices adoption antipatterns	[31]
G2	Microservices antipatterns and pitfalls	[32]
G3	Pitfalls & Challenges Faced During a Microservices Architecture Implementation	[33]
G4	MICROSERVICE ANTI-PATTERN: THE SERVICE MESH	[34]
G5	The Seven Deadly Sins of Microservices (Redux)	[35]
G6	Microservice Patterns & Antipatterns	[36]
G7	Microservices Anti-patterns: A Taxonomy	[49]
G8	Microservice Antipatterns: The Queue Explosion	[37]
G9	Microservices Anti patterns	[38]
G10	Disasters I've seen in a microservices world	[42]
G11	Managing Application Dependencies in Distributed Architectures	[43]
G12	How We Reorganize Microservices Platform Team	[44]
G13	Seven Ways to Fail at Microservices	[47]

TABLE A.2
SELECTED STUDIES IN ACADEMIC LITERATURE

ID	Title	Year	Reference
A1	Does migrating a monolithic system to microservices decrease the technical debt?	2020	[14]
A2	Fine-Grained Access Control for Microservices	2018	[15]
A3	Improving Agility by Managing Shared Libraries in Microservices	2020	[16]
A4	Verification of Microservices Using Metamorphic Testing	2019	[17]
A5	Microservices Migration in Industry: Intentions, Strategies, and Challenges	2019	[18]
A6	Microservice Architecture in Reality: An Industrial Inquiry	2019	[19]
A7	Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality	2019	[20]
A8	Assuring the Evolvability of Microservices: Insights into Industry Practices and Challenges	2019	[21]
A9	From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture	2017	[22]
A10	ANT-Man: Towards Agile Power Management in the Microservice Era	2020	[23]
A11	On the definition of microservice bad smells	2018	[24]
A12	Microservices: Architecting for Continuous Delivery and DevOps	2018	[25]
A13	Microservices: Migration of a Mission Critical System	2018	[26]
A14	Anti-Patterns for Multi-Language Systems	2019	[27]
A15	Architectural Technical Debt in Microservices: A Case Study in a Large Company	2019	[28]
A16	Fifer: Tackling Resource Underutilization in the Serverless Era	2020	[29]
A17	The Hidden Cost of Backward Compatibility: When Deprecation Turns into Technical Debt - an Experience Report	2020	[30]
A18	Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study	2021	[39]
A19	Promises and challenges of microservices: an exploratory study	2021	[40]
A20	An Empirical Study on Microservice Software Development	2021	[41]
A21	Serverless: From Bad Practices to Good Solutions	2022	[46]