Fostering Microservice Maintainability Assurance through a Comprehensive Framework

Amr S. Abdelfattah
Computer Science Department, Baylor University
Waco, TX, USA
amr_elsayed1@baylor.edu
Supervisors: Tomas Cerny & Eunjee Song

***Abstract*—Cloud-native systems represent a significant leap in constructing scalable, large systems, employing microservice architecture as a key element in developing distributed systems through self-contained components. However, the decentralized nature of these systems, characterized by separate source codes and deployments, introduces challenges in assessing system qualities. Microservice-based systems, with their inherent complexity and the need for coordinated changes across multiple microservices, lack established best practices and guidelines, leading to difficulties in constructing and comprehending the holistic system view. This gap can result in performance degradation and increased maintenance costs, potentially requiring system refactoring. The main goal of this project is to offer maintainability assurance for microservice practitioners. It introduces an automated assessment framework tailored to microservice architecture, enhancing practitioners' understanding and analytical capabilities of the multiple system perspectives. The framework addresses various granularity levels, from artifacts to constructing holistic views of static and dynamic system characteristics. It integrates diverse perspectives, encompassing human-centric elements like architectural visualization and automated evaluations, including coupling detection, testing coverage measurement, and semantic clone identification. Validation studies involving practitioners demonstrate the framework's effectiveness in addressing diverse quality and maintainability issues, revealing insights not apparent when analyzing individual microservices in isolation.**

***Index Terms*—Microservice Architecture, Microservice Visualization, Maintainability, Metrics, Software Architecture Reconstruction, Viewpoints**

I. Introduction

Cloud-native design emerged as the mainstream direction for decentralized systems seeking scalability and high performance under heavy workloads. The adoption of microservice architecture for such direction is crucial for fully harnessing the advantages of cloud computing [1]. Microservices, with their self-contained nature, enhance management and deployment efficiency, making them highly favored in enterprise software systems. The microservice architecture, once considered cutting-edge, has now become the standard, with industry giants like Amazon and Netflix leading the way. These paradigms have profoundly transformed application development, emphasizing scalability and efficiency.

Problem Statement: The problem statement revolves around system maintainability. In particular, it considers challenges stemming from comprehension of the overall system, attributed to poor design decisions and heightened system complexity. Software systems undergo transformations throughout their lifecycle, whether to incorporate new features, adapt to different environments, address bugs, and more. However, these changes often pose significant challenges. As highlighted by Bass et al. [2], the software development community is grappling with the realization that about 80% of the total cost of a typical software system is incurred after its initial deployment to modify and maintain the system. Consequently, a significant portion of the systems that developers engage with is in this post-deployment phase. Many software practitioners work within the constraints of the existing architecture and codebase, with limited opportunities for new development.

Moreover, managing a complex architecture, characterized by its dynamic, distributed, and expansive nature, poses significant challenges. Independent decision-making by teams on parts of the system, coupled with the evolving nature of the system, can lead to ripple effects where changes in one part impact another, complicating management processes. Additionally, maintaining up-to-date documentation for numerous interconnected services, each with multiple dependencies, presents substantial challenges in management and maintenance. These efforts may result in downtime and necessitate a comprehensive overhaul of the entire system.

Research Objective: The objective is to introduce and implement an automated assessment framework for microservice systems. This objective is fueled by building comprehensive foundations especially with the absence of comprehensive guidelines governing microservices practices. This framework aims to enhance analytical abilities and reasoning in architectural design, identify early indicators of system design degradation, and offer testability measurements for the system.

Paper Organization: The rest of this paper is organized as follows: Section II covers the background and objectives with research questions (RQs). Section III details the methodology and prior results. Section IV discusses the development of the maintainability framework. Section V outlines the research contributions and conclusion.

II. Background and Objective

Microservices, foundational to cloud-native applications, are modular, autonomous units that enhance agility and resource utilization by decomposing functionalities [3], [4]. They interact via network protocols and message passing, improving system modularity, scalability, and maintainability. Maintainability, a key quality attribute, measures how effectively and efficiently a product can be modified by its maintainers [2]. According to ISO 25010 [5], it includes essential dimensions like modularity, reusability, analyzability, modifiability, and testability.

The main objective is to design and implement an automated assessment framework for the maintainability of microservice systems. To achieve this, several RQs must be explored, each contributing to the framework and supporting system maintainability.

RQ$_1$ *What dependencies exist within microservices-based systems?*

In microservices-based systems, dependencies become distributed, heterogeneous, less visible, and harder to track. The challenge is compounded by the absence of comprehensive guidelines and catalogs for identifying and defining these dependencies. Such guidelines are crucial for enhancing system maintainability. *This RQ aims* to develop a dependency taxonomy to offer a comprehensive understanding of these dependencies, their impact, and their categorization.

RQ$_2$ *How to construct system-centric views of microservice architecture?*

The decentralized nature of microservices and their dependencies presents a challenge wherein alterations in one part of the architecture can yield widespread effects, both explicitly and implicitly. Teams frequently enact these changes without possessing a comprehensive view of the entire system. Moreover, the isolated nature of teams, each operating within separate sandboxes, fosters independent decision-making, potentially influencing the overall architecture. *This RQ aims* to develop holistic and centric views of the system architecture that encompass various perspectives of dependencies.

RQ$_3$ *How to Implement automatic testability and quality assessment methodologies on the system?*

A critical challenge facing microservice systems is the absence of tailored quality assessment techniques. Existing methodologies often fail to address the unique characteristics and complexities inherent in microservice architectures [6], [7]. This deficiency hampers the ability of development teams to effectively evaluate the maintainability of their microservice-based applications, thus hindering their overall quality assurance efforts. *This RQ aims* to employ holistic views to formulate and automate assessment metrics and techniques. It seeks to adopt a holistic approach that considers various facets of the system architecture and its operation to create robust and efficient evaluation methodologies.

RQ$_4$ *How can a tailored visualization methodology be designed to effectively communicate the constructed views to practitioners?*

The extensive information encompassed within microservice architecture and its holistic views presents a challenge in effectively conveying this wealth of data to practitioners. Furthermore, the absence of visualization tools specifically tailored for microservices views raises questions about the adequacy of traditional visualization techniques in capturing microservice perspectives. *This RQ aims* to design and validate a visualization technique specifically tailored for depicting representations of microservices. It aims to develop a visualization approach that effectively captures the intricacies and dependencies inherent in microservice architectures, providing practitioners with intuitive and insightful visualizations to aid in their understanding and reasoning processes.

III. Methodology and Prior Results

Different research questions necessitate distinct methodologies to address them. Employing diverse methodologies enhances the breadth of outcomes and validates their robustness.

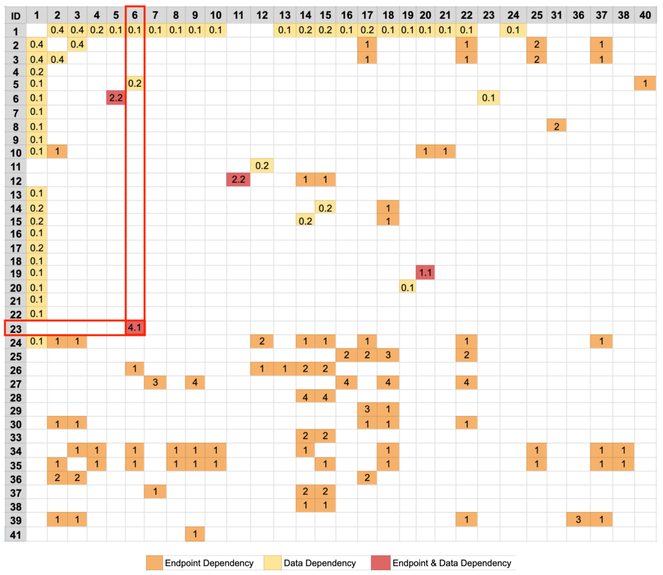
Regarding **RQ**$_1$, it embarks on the ambitious endeavor of conducting a comprehensive study to grasp the foundational issues stemming from the core concept of dependencies. This study is centered on the creation of a varied dependency taxonomy, which amalgamates insights from existing literature, tackles challenges to comprehend existing dependencies, and draws upon various expertise to elucidate multifaceted dependency dimensions. It distinguishes between root causes and symptomatic occurrences, thus reducing ambiguity during identification. *To tackle this RQ*, two methodologies are utilized: Systematic Literature Review (SLR) [8] and Open and Axial coding [9], [10]. The SLR collects literature evidence to understand current microservice dependencies. Then, Open and Axial coding validate and categorize these dependencies with expert collaboration. These methodologies are evaluated for their systematic approach and efforts to reduce bias by involving external practitioners.

Prior results pertaining to this RQ include a discussion paper on the correlation between dependencies and system maintainability, as referenced in [11]. Furthermore, the dependency taxonomy (under review) reveals both implicit and explicit types of dependencies. Explicit dependencies occur when one microservice directly calls another via a REST interface [12]. Similarly, data dependencies arise when multiple data entities are shared between microservices within a bounded context. Implicit dependencies arise through semantic clones, where multiple microservices share similar business logic.

For **RQ**$_2$, the aim is to leverage the dependency taxonomy and constructed guidelines to build system-centric views of microservice-based systems. This involves creating multidimensional perspectives (viewpoints) based on various dependency artifacts, which form the cornerstone of the research endeavor, utilizing the established guidelines to formulate subsequent research questions and provide practitioners with a comprehensive system view. *To address this RQ*, the Software Architecture Reconstruction (SAR) [13] methodology is employed to automatically generate these comprehensive views.

Additionally, a review methodology is utilized to develop a roadmap for generating different representations from the system artifacts. A review has been conducted to introduce this roadmap [3] for constructing the necessary viewpoints. Both static and dynamic analyses are employed to capture the multiple dimensions of the system. Prototypes and case studies are then developed to assess the effectiveness of the method and the completeness of the generated viewpoints.

Prior results have been derived from endpoint dependencies and data dependencies extracted using static analysis techniques. These results are presented in dependency matrix representations as outlined in [14]. These matrices offer a central viewpoint illustrating the behavior of dependencies across dimensions, such as the Service Dependency Matrix (SDM) depicted in Fig. 1 (generated from the TrainTicket benchmark [15]). For example, considering the cell at position (row: 23, column: 6) in the SDM, with a value of 4.1, it signifies that the `ts-admin-user-service` microservice (ID 23) has made four calls to the `ts-user-service` microservice (ID 6), with one shared entity (`UserDto`) between them. These dependencies underscore their interconnected nature and the significance of their mutual interaction.



ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	31	36	37	38	40
1		0.4	0.4	0.2	0.1	0.1	0.1	0.1	0.1	0.1			0.1	0.2	0.2	0.1	0.2	0.1	0.1	0.1	0.1	0.1		0.1						
2	0.4		0.4														1					1			2			1		
3	0.4	0.4															1					1			2			1		
4	0.2																													
5	0.1					0.2																								1
6	0.1				2.2																		0.1							
7	0.1																													
8	0.1																									2				
9	0.1																													
10	0.1	1																		1	1									
11												0.2																		
12											2.2			1	1															
13	0.1																													
14	0.2														0.2			1												
15	0.2													0.2				1												
16	0.1																													
17	0.2																													
18	0.1																													
19	0.1																			1.1										
20	0.1																		0.1											
21	0.1																													
22	0.1																													
23						4.1																								
24	0.1	1	1									2		1	1		1					1						1		
25																2	2	3				2								
26						1						1	1	2	2															
27							3		4							4		4				4								
28														4	4															
29																	3	1												
30		1	1														1	1				1								
33														2	2															
34			1	1		1		1	1	1				1				1							1			1	1	
35		1		1		1		1	1	1					1			1							1			1	1	
36		2	2														2													
37							1							2	2															
38														1	1															
39		1	1																			1					3	1		
41									1																					

🟧 Endpoint Dependency 🟨 Data Dependency 🟥 Endpoint & Data Dependency

Fig. 1: Service Dependency Matrix (SDM). This matrix is sourced from Abdelfattah et al. (2023) [14].

For **RQ**$_3$, two main tasks are addressed: firstly, creating a comprehensive catalog of microservice bad practices to establish benchmarks for assessing architecture maintainability; secondly, leveraging the developed holistic views to devise and automate assessment metrics and techniques.

Transferring design patterns across various software architectures is challenging, as not all patterns fit universally, leading to anti-patterns. To gauge maintainability, current systems are assessed for anti-patterns, often using tools like SonarQube [16]. However, SonarQube evaluates individual applications and cannot identify microservice smells or detect anti-patterns across codebases. This highlights the need to explore and identify anti-patterns specific to microservices for effective system maintenance. *To achieve this*, this research utilizes the Triatry systematic literature review [17] in conjunction with the Open and Axial coding methodologies to construct a cohesive and exhaustive catalog focused on microservice ecosystem anti-patterns. Additionally, it leverages the developed viewpoints and integrates dynamic data to assess the system by analyzing runtime behavior and computing assessment metrics such as endpoint test coverage metrics.

Prior results introduced a comprehensive anti-pattern catalog in [18], comprising 5 main categories and 58 individual anti-patterns. This catalog engages experts from the field to ensure consistency and merge similar anti-patterns published under different names or with slightly varied definitions. It lays the groundwork for developing assessment techniques to identify and address bad practices within microservice systems. Regarding assessment metrics, this research has delivered End-to-End (E2E) and API test coverage metric calculations, as detailed in [19], [20]. It presents three metrics for endpoint test coverage calculation in microservice systems. Moreover, it encompasses additional metrics aimed at gauging the impact of evolutionary changes across various components of the system.

For **RQ**$_4$, the focus lies on determining the suitable design approach to effectively convey the complex information extracted from the microservice ecosystem. The challenge lies in visualizing the holistic view in a more engaging and interactive manner, rather than relying solely on static dependency matrices. This prompts the question of whether conventional methodologies suffice for representing this information or if a tailored visualization methodology should be introduced. *To tackle this RQ*, we initiated by conducting a thorough literature review to assess the state of the art in visualization within this domain. Subsequently, we devised and developed a visualization methodology, followed by a series of controlled experiments involving practitioners to evaluate the understandability of microservice dependency behavior through both conventional and tailored visualization approaches.

Prior results performed multiple reviews addressing visualization in the literature [21], [22], which highlighted a shortage in microservice visualization methods, necessitating a tailored design to encompass its unique perspectives. We then presented the design outline of the visualization methodology experiment in [23], which involved implementing a tool named Microvision utilizing Augmented Reality (AR) medium to address the challenge of limited rendering space in traditional visualization, as detailed in [24]. For assessment purposes, we conducted controlled experiments comparing Microvision with conventional 2D-graph-based visualization tools, with the protocol and results analysis published in [25]. The 2D tool uses rectangular boxes and arrows to present microservice dependency graphs, similar to commercial and open-source tools. In contrast, the Microvision tool renders a 3D model with cubes and line connectors, displaying call information via popups. Both visualizations convey the same information but offer distinct display and interaction features for different needs. This experiment involved systems of two sizes (small and large) and participants with varying expertise (novice and expert). The findings showed that AR effectively aids in understanding microservice architecture, enabling novice practitioners to grasp system dependencies like experienced users. While 2D tools clearly visualized dependencies in small systems, Microvision outperformed in visualizing large systems with scalability in mind.

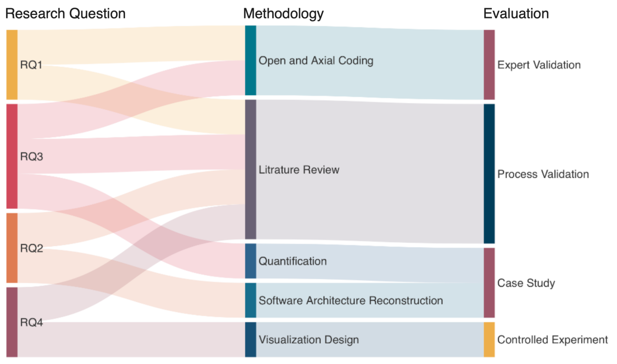


Fig. 2: Methodologies aligned with research questions.

As depicted in Fig. 2, each research question employs a distinct literature review methodology tailored to its specific tasks. RQ_1 and RQ_3 utilize the Open and Axial coding methodology to refine the data extracted from the literature review. Regarding evaluation methodologies, this study assesses each method based on its inherent nature. The literature review undergoes validation through adherence to process guidelines, while the Open and Axial coding process is validated through collaboration with experts. Case studies are employed to validate the Software SAR process and quantification metrics. Additionally, controlled experiments are utilized to evaluate the effectiveness of the visualization design methodology.

IV. Framework Construction

These RQs' perspectives form the basis for developing an assessment framework to allow analyzers to operate based on extracted components and constructed holistic views.

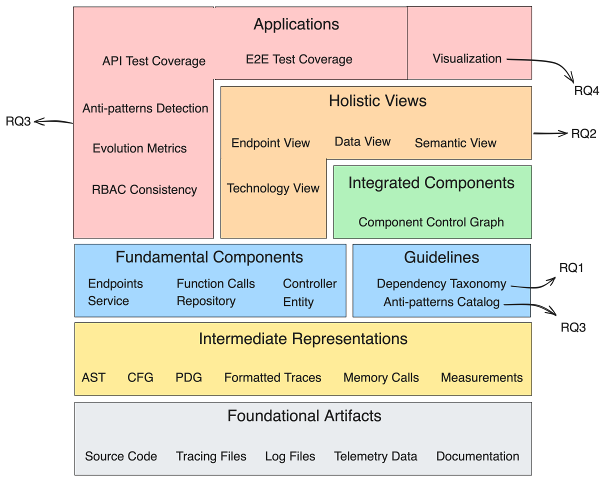


Fig. 3: The proposed Microservice Assessment Framework

This framework, as depicted in Fig. 3 is centered around the construction of the system holistic view and derive automated application to provide assessment for the system quality perspectives. This framework considers various system granularities and perspectives, combining human-centric aspects like architectural visualization with automated evaluation facets. This framework adheres to the principles outlined in layers-based guidelines [2], which promote the use of layers to offer guidance and flexibility in identifying architectural aspects and determining the views to be generated. This six-layered framework originates from the microservice system artifacts and culminates at the application layer, the epicenter of assessment objectives. The application layer (both quality assessment perspective (RQ_3) and visualization perspective (RQ_4)) harmonizes with one of three granularity component layers. At the fundamental components layer, the system components are extracted from the constructed intermediate representation, an outcome of the intricate static and dynamic analysis extraction processes. Above it lies the integrated components [26] layer, a flow of interconnected components within individual microservices. Contrarily, the holistic views (RQ_2) layer orchestrates the interconnected components and flows across multiple microservices, constructing a comprehensive view of the entire system. Moreover, the guidelines (dependency taxonomy (RQ_1) and anti-pattern catalog (RQ_3)) serve as valuable resources to augment the understanding and application of microservices practices. This concerted effort to compile such essential references directly fuels the fundamental component of this framework.

The introduced framework provides flexibility to the application layer, allowing diverse applications to assess different system perspectives. The outcomes demonstrate that this framework effectively addresses issues through a holistic viewpoint, uncovering insights that might be missed when analyzing individual microservices in isolation.

V. Contribution and Conclusion

This work introduces a comprehensive framework for microservices systems, enhancing system maintainability and analytical capabilities. The framework offers techniques for effective system modifications, ensuring testability, and improving modularity and reusability to maintain manageability.

In addition to practical applications, this work provides valuable resources to the community. These include multiple publications detailing the framework's construction, comprehensive catalogs for anti-patterns and dependency taxonomy, datasets with dependency measurements, anti-pattern definitions, testing suites, and a visualization tool for holistic service views, integrated as a plugin with development tools.

The framework paves the way for future research, such as exploring new holistic perspectives from technological and operational standpoints, broader applications like interactive documentation, and incorporating evolutionary aspects through co-change analysis.

Funding

This material is based upon work supported by the National Science Foundation under Grant No. 2409933.

References

[1] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual understanding of microservice architecture: Current and future directions," *SIGAPP Appl. Comput. Rev.*, vol. 17, no. 4, pp. 29–45, Jan. 2018.

[2] L. Bass, P. Clements, and R. Kazman, *Software architecture in practice*, 3rd ed. Addison-Wesley Professional, 2003.

[3] A. S. Abdelfattah and T. Cerny, "Roadmap to reasoning in microservice systems: A rapid review," *Applied Sciences*, vol. 13, no. 3, p. 1838, 2023.

[4] Amazon, "What is cloud native? cloud native applications explained," https://aws.amazon.com/what-is/cloud-native/, 2024, accessed: 2024-02-01.

[5] "Iso/iec 25010," https://iso25000.com/index.php/en/iso-25000-standards/iso-25010, accessed: May 14, 2024.

[6] I. Ghani, W. M. Wan-Kadir, A. Mustafa, and M. I. Babir, "Microservice testing approaches: A systematic literature review," *International Journal of Integrated Engineering*, vol. 11, no. 8, pp. 65–80, 2019.

[7] P. Jiang, Y. Shen, and Y. Dai, "Efficient software test management system based on microservice architecture," in *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 10. IEEE, 2022, pp. 2339–2343.

[8] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and software technology*, vol. 55, no. 12, pp. 2049–2075, 2013.

[9] S. H. Khandkar, "Open coding," *University of Calgary*, vol. 23, no. 2009, p. 2009, 2009.

[10] J. Kendall, "Axial coding and the grounded theory controversy," *Western journal of nursing research*, vol. 21, no. 6, pp. 743–757, 1999.

[11] T. Cerny., M. Chy., A. Abdelfattah., J. Soldani., and J. Bogner., "On maintainability and microservice dependencies: How do changes propagate?" in *Proceedings of the 14th International Conference on Cloud Computing and Services Science - CLOSER*, INSTICC. SciTePress, 2024, pp. 277–286.

[12] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc.", 2011.

[13] G. Y. Guo, J. M. Atlee, and R. Kazman, "A software architecture reconstruction method," in *Software Architecture: TC2 First Working IFIP Conference on Software Architecture (WICSA1) 22–24 February 1999, San Antonio, Texas, USA*. Springer, 1999, pp. 15–33.

[14] A. S. Abdelfattah and T. Cerny, "The microservice dependency matrix," in *European Conference on Service-Oriented and Cloud Computing*. Springer Nature Switzerland Cham, 2023, pp. 276–288.

[15] FudanSELab, "Train Ticket Wiki," https://github.com/FudanSELab/train-ticket/wiki, Accessed: 2024.

[16] "Sonarqube," https://www.sonarsource.com/products/sonarqube/, accessed: May 14, 2024.

[17] B. Kitchenham, R. Pretorius, D. Budgen, O. P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering–a tertiary study," *Information and software technology*, vol. 52, no. 8, pp. 792–805, 2010.

[18] T. Cerny, A. S. Abdelfattah, A. Al Maruf, A. Janes, and D. Taibi, "Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study," *Journal of Systems and Software*, vol. 206, p. 111829, 2023.

[19] A. S. Abdelfattah, T. Cerny, J. Yero, E. Song, and D. Taibi, "Test coverage in microservice systems: An automated approach to e2e and api test coverage metrics," *Electronics*, vol. 13, no. 10, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/10/1913

[20] A. S. Abdelfattah, T. Cerny, J. Y. Salazar, A. Lehman, J. Hunter, A. Bickham, and D. Taibi, "End-to-end test coverage metrics in microservice systems: An automated approach," in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2023, pp. 35–51.

[21] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, "Microservice architecture reconstruction and visualization techniques: A review," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2022, pp. 39–48.

[22] K. Burgess, D. Hart, A. Elsayed, T. Cerny, M. Bures, and P. Tisnovsky, "Visualizing architectural evolution via provenance tracking: a systematic review," in *Proceedings of the Conference on Research in Adaptive and Convergent Systems*, 2022, pp. 83–91.

[23] A. S. Abdelfattah, "Microservices-based systems visualization: student research abstract," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 1460–1464.

[24] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, "Microvision: Static analysis-based approach to visualizing microservices in augmented reality," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2022, pp. 49–58.

[25] A. S. Abdelfattah, T. Cerny, D. Taibi, and S. Vegas, "Comparing 2d and augmented reality visualizations for microservice system understandability: A controlled experiment," in *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*. IEEE, 2023, pp. 135–145.

[26] A. S. Abdelfattah, A. Rodriguez, A. Walker, and T. Cerny, "Detecting semantic clones in microservices using components," *SN Computer Science*, vol. 4, no. 5, p. 470, 2023.