IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 December 2023, accepted 5 February 2024, date of publication 9 February 2024, date of current version 16 February 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3365079

RESEARCH ARTICLE

Evolution of Microservices Identification in Monolith Decomposition: A Systematic Review

IDRIS OUMOUSSA AND RAJAA SAIDI
SI2M Laboratory, National Institute of Statistics and Applied Economics (INSEA), Rabat 10100, Morocco
Corresponding author: Idris Oumoussa (ioumoussa@insea.ac.ma)

ABSTRACT Modernizing monolithic systems through microservices architectures (MSAs) promises significant benefits in terms of scalability, agility, and cloud adoption. However, this transition hinges on accurate and efficient microservices identification, a complex area of research still in its evolution. This systematic literature review delves into this challenge by exploring three critical questions: First, we examine how the field of microservices identification has evolved over time, analyzing publication trends, categorizing existing research, and mapping out different research objectives and methodologies employed. Second, we dive into the current state-of-the-art, showcasing cutting-edge methodologies and tools developed to address microservices identification challenges. We highlight promising approaches while identifying potential limitations. Third, we unveil both existing hurdles and future challenges in this domain, painting a comprehensive picture of the obstacles and opportunities that lie ahead. Our findings illuminate key areas demanding further attention, including the need for more automated and accurate identification tools, standardized evaluation benchmarks, and a deeper understanding of the human factors involved in successful transitions. By addressing these critical gaps, we aim to pave the way for smoother and more effective modernization of monolithic systems through microservices adoption.

INDEX TERMS Microservices, microservices architecture, microservices identification, monolith application decomposition, monolith to microservices migration.

I. INTRODUCTION

In the face of increasingly complex software systems and a relentless drive for agility, monolithic architectures have begun to reveal their limitations. Microservices architectures, with their emphasis on independent, self-contained services, offer a compelling alternative, promising enhanced scalability, faster deployment cycles, and improved maintainability [1], [2], [3]. However, the transition to this fragmented paradigm is not without its challenges. One of the most formidable is the decomposition of existing monoliths into cohesive microservices.

Effectively identifying microservice boundaries and functional responsibilities within a monolithic software system is a critical task, yet often proves elusive. Various techniques have emerged to assist in this process, analyzing features, dependencies, and execution patterns to potentially carve out well-defined microservices [4], [5], [6], [7], [8]. Despite these advancements, a comprehensive understanding of the strengths, weaknesses, and ongoing challenges of existing decomposition strategies remains elusive.

This systematic literature review aims to bridge this knowledge gap. Through a rigorous methodology, we systematically compile, analyze, and synthesize research contributions on monolith decomposition, with a specific focus on techniques for microservices identification. Our exploration probes into the research objectives, evaluation methods, and persistent challenges that characterize this domain, seeking to establish a robust classification of decomposition approaches and illuminate avenues for further refinement.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudia Raibulet.

Despite burgeoning interest, microservices identification remains in its infancy, grappling with several critical limitations: Fragmented data collection and analysis techniques hinder the extraction of crucial features from monolithic software, impeding effective identification. Scarce head-to-head comparisons of existing methods obscure the most effective approaches for different scenarios. Universally accepted quality metrics for potential microservice candidates remain elusive, making objective assessment a challenge. Finally, a lack of integrated tools to seamlessly support the entire identification pipeline, from data gathering to candidate refinement, further complicates the process. This systematic literature review aims to shed light on these obstacles and chart a path towards more precise and reliable microservices identification techniques.

The remaining sections of this paper are organised as follows. Section II defines microservices architecture and microservices identification. Section III describes the methodology used to locate the chosen works. Section IV focuses on the objectives, methods, and evaluations utilised in microservices identification research. Section V provides a summary of the state of the art in microservices identification research. Section VI identifies open microservices identification challenges that are either partially or completely unresolved by current research. The validity of this paper is described in Section VII. Finally, Section VIII concludes the paper.

II. PRELIMINARIES

This section briefly introduces the concept of microservice architectures and provides an overview of microservices identification.

A. MICROSERVICES ARCHITECTURES

As a modern computing paradigm that has gained popularity in software engineering. MSA breaks down traditional monolithic applications into fine-grained, independent services that can be designed, tested, and deployed individually [9]. It enhances application scalability, simplifies partnerships and service integration across well-defined interfaces [4].

Microservices are characterized by their lightweight nature, where each service has discrete responsibilities and collaborates with similar services through well-defined interfaces. They communicate using lightweight protocols like asynchronous message buses [10]. Microservices can be developed independently, utilizing various frameworks, programming languages, and resources. Functional decomposition of applications is a key aspect of microservices, allowing the construction of applications or services at a higher level by combining various services. Fine granularity and loose coupling are essential properties of microservices [11].

Furthermore, deploying a single business capability per microservice enables their use across diverse applications and domains. The primary attributes distinguishing the microservices architectural style from both monolithic and service-oriented architectures are the reduced size, scalability, and autonomy of each component composing a system.

B. FROM MONOLITHIC TO MICROSERVICES

The inherent complexity of monolithic architectures, characterized by tightly coupled components, poses significant challenges for maintainability and scalability. As software systems evolve, these challenges become increasingly burdensome, hindering both development and deployment agility. Microservices with their independent and cohesive services, offer a compelling alternative, enabling the construction of intricate applications through modularity and simplified integration. However, transitioning to this paradigm entails a multifaceted process involving meticulous migration procedures, sophisticated microservices extraction techniques, and rigorous service quality assessment.

At the core of this transition lies the critical phase of microservices identification. The success of the newly composed architecture hinges on the selection of optimal services, characterized by fine granularity to facilitate agile change management, convenient maintenance, and effortless reuse. Identifying these services necessitates a systematic approach, encompassing thorough dissection of the existing system into distinct functional units, precise definition of service boundaries, and the strategic application of both static and dynamic analysis techniques. While essential, the task of pinpointing optimal microservices remains intricate, demanding a methodological approach potentially augmented by automated detection techniques to navigate the complexities with greater efficiency and accuracy.

III. METHODOLOGY

In our paper, we employed a methodical, organized, and systematic approach to produce a survey on the topic of microservices identification. Our approach was based on the recommendations and guidance provided by Kitchenham and Charterss [12] and Petersen et al. [13].

A. RESEARCH QUESTIONS

This systematic literature review comprehensively analyzes the current state of research on microservices identification, serving as a reference for existing techniques and identifies unresolved research questions. To achieve this, we have developed the following research questions (RQs):

- RQ1: *How has the area of study on microservices identification evolved ?* We aim to examine published studies on microservices identification, summarize and classify them, describe their objectives, evaluate methodologies used, and discuss trends. We present our findings for this RQ in Section IV.

- RQ2: *What is the current state-of-the-art in microservices identification research ?* We will present state-of-the-art methodologies and tools proposed for addressing microservices identification challenges. We present our findings in Section V.

- RQ3: *What are the current and potential challenges associated with microservices identification ?* Our goal is to identify both existing and future challenges in microservices identification research. We present our findings in Section VI.

B. LITERATURE REPOSITORY SELECTION

Search string used in this study is designed to be generic and simple. It is constructed based on search terms concerned with population and intervention as suggested by Petticrew and Roberts in [14]. Population refers to the application area which is microservices and monolith where intervention is identification, decomposition and migration. Accordingly, final adopted search string is:

(''monolith'' OR ''existing'' OR ''legacy'')
AND
(''microservices'' OR ''micro-services'')
AND
(''identification'' OR ''decomposition'' OR ''extraction'')

To establish the selection criteria for online literature repositories, we consulted prior state-of-the-art literature reviews in software engineering [15], [16]. We first picked publications from the following technical publishers:

- ACM Digital Library
- Elsevier Science Direct
- IEEE Xplore Digital Library
- Springer Online Library
- Wiley Online Library

We enhanced our literature search for microservices identification studies by conducting a specific search on Google Scholar using the keyword ''microservices identification''. This strategy aimed to expand our coverage of technical publications,[1] we utilised ''microservices identification'' and ensure the inclusion of a wide range of relevant articles. Additionally, we performed manual searches of the references in our initial selection using forward and backward snowballing techniques, leveraging Google Scholar to identify additional works related to our research objectives.

C. LITERATURE SEARCH AND SELECTION

We conducted literature searches in our predefined repositories using the exact search phrase ''microservices identification''. The results, as shown in Table 1, display the total number of publications found in each library. We manually filtered these results, retaining only those that met the following criteria:

- Studies must be written in English
- Studies must be related to computer science or software engineering
- Studies should have a relation to microservices identification
- Studies must not be a Master or PhD thesis

[1]Repository of our primary studies and classifications: https://github.com/Ioumoussa/MicroservicesIdentificationSurveyPapers

TABLE 1. Number of studies returned by each repository.

Repository	Search results
IEEE Xplore	50
ACM Digital Library	46
Springer Link	674
Science Direct	252
Wiley Online Library	39
Total	1061

- Studies must be fully available from one or more online library

A flowchart of our publication selection process can be found in figure 1. Initially, we identified 35 publications through our filtering process. An additional 133 papers were included after reviewing the references of selected papers, as some relevant publications may have been missed due to terminology variations (e.g., ''microservices extraction'' instead of ''microservices identification''). In total, we selected 168 articles (or *primary studies*) for this survey based on our initial search results and references that met our filtering criteria.

D. OVERVIEW OF PRIMARY STUDIES

The publications in this study are distributed across various venues, with variations in prominence. The ''Journal of Systems and Software'' stands out as the primary journal, hosting nine papers related to microservices identification, while ''IEEE Access'' also contributes significantly with six journal publications. Figure 2 illustrates that most microservices identification publications are conference papers, followed by journal papers, while books make up a minimal portion.

We classified the 168 publications in our sample into five contribution types using an open-card sorting approach. These types were determined based on author and publisher keywords, publication venue information, and our subjective judgment. The five contribution types are:

1) New Tools and Techniques: This category encompasses publications introducing novel tools and techniques specifically designed for microservices identification.
2) Empirical Studies: Focused on data analysis and evidence-based findings, this category includes publications presenting empirical evaluations of existing methodologies or novel approaches.
3) Tools and Technique Proposals: While laying out innovative tools or techniques, publications belonging to this category lack implementations or experimental results.
4) Surveys: These publications offer systematic analyses of multiple existing works within the field of microservices identification.
5) Datasets: This category comprises publications that contribute and share novel datasets specifically geared towards advancing future research in microservices identification.

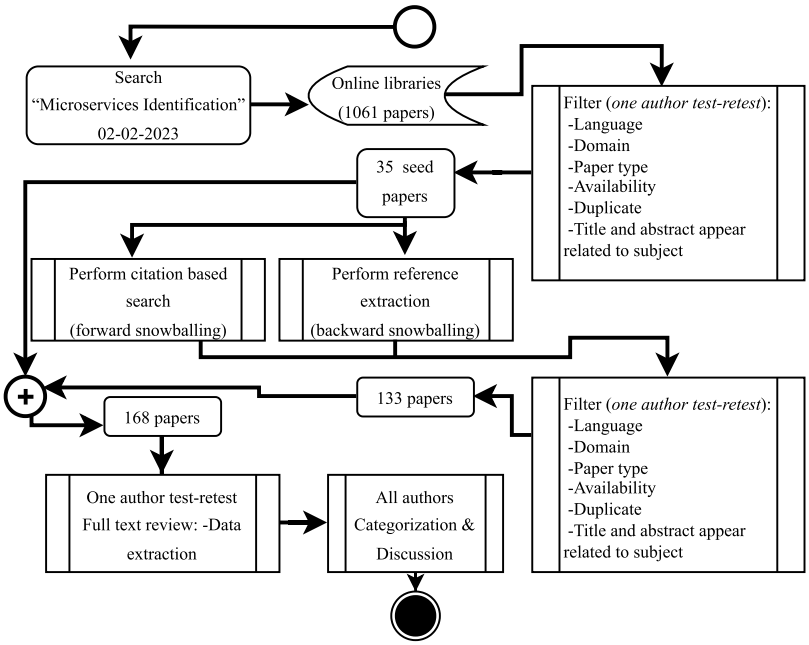

FIGURE 1. Our paper selection process.

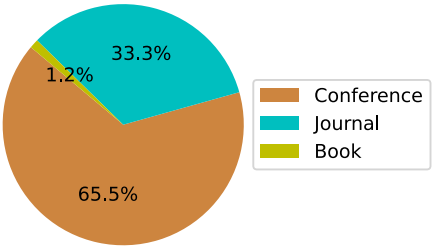

FIGURE 2. Overall venue distribution.

TABLE 2. Microservices identification publication contribution types.

Main contribution	Papers
New Tools and Techniques	54
Empirical Studies	41
Tools and Technique Proposals	21
Surveys	48
Datasets	4

FIGURE 3. Keyword cloud of primary research.

The detailed classification results are available in Table 2. To assess the significance of our primary investigations on microservices identification, we employed Word Clouds as recommended in [17]. Figure 3 displays the most prevalent terms in titles and abstracts, such as microservice, architecture, and service. Terms related to identification methods like clustering, classification, and similarity are also common. The Word Cloud underscores the importance of the break-down process in identifying microservices and the need for effective extraction and classification procedures.

IV. EVOLUTION OF MICROSERVICES IDENTIFICATION RESEARCH

In response to RQ1, which addresses the evolution of microservices identification research, we provide a two-part response: (1) research goals in microservices identification and (2) microservices identification research evaluation.

A. MICROSERVICES IDENTIFICATION RESEARCH GOALS

In response to the first part of RQ1, we have identified various research objectives within the field of microservices identification. This research primarily focuses on decomposing monolithic applications into microservices, emphasizing the importance of identifying appropriate microservice boundaries to ensure loose coupling, maintainability, and scalability. Achieving these objectives often involves empirical studies to observe monolithic application behavior and identify potential candidate services, which are then validated using automated or manual techniques. The overarching goal of microservices identification research is to develop tools and techniques that facilitate the migration process.

This includes identifying service dependencies, pinpointing performance bottlenecks, and ensuring data consistency across microservices. We have categorized research contributions into types such as Datasets, Empirical Studies, Tools and Techniques Proposals, Surveys, and New Tools and Techniques to gain insights into primary research goals and trends. These objectives may encompass improving the precision and efficiency of microservices identification, addressing challenges in microservices integration, and devising innovative migration strategies.

1) NEW TOOLS AND TECHNIQUES

The landscape of microservices identification tools and techniques brims with innovation, each contributing unique approaches to tackle the multifaceted challenges of transitioning from monolithic architectures. Analyzing codebases [18], [19] is a common approach, while techniques like automation [20], [21] and even cutting-edge methods like genetic algorithms [22] and neural networks [23] are emerging. Some tools prioritize simplifying microservices for improved maintainability [24], while others focus on preserving compatibility with existing systems [25]. This vibrant research realm paves the way for streamlined migration, empowering organizations to reap the benefits of microservices, while avoiding pitfalls like complex architectures and intricately tangled service dependencies. Ultimately, these innovative tools and techniques equip developers with the power to efficiently dissect monolithic behemoths and sculpt them into well-defined, independent microservices, unlocking the true potential of this transformative architectural paradigm.

Answers of RQ1: New tools and techniques typically aim to assist with microservices identification. They aim to resolve issues that can arise for developers during this migration, such as tools for microservices migration, or to reduce the development effort needed for identifying and managing microservices.

2) EMPIRICAL STUDIES

Empirical investigations represent the third-largest category of publications in microservices identification. These studies include data-mining research, case studies, and user studies.

- *Data-mining Studies:* These studies use large datasets to identify problems and assess their impact on microservices. They examine issues such as microservices failure, system evolution, and compatibility [26], [27], [28], [29].
- *Case Studies:* Case studies typically focus on a small number of systems, often fewer than ten. The findings from these studies are specific to the systems under investigation and address various research objectives and challenges. Researchers have explored a wide range of topics, including the impact of microservices identification on system users [7], the influence of technical debt on the success of migrating to microservices [30], and the factors contributing to the long-term success of microservices frameworks [31].
- *User Studies:*
These papers rely heavily on human responses to address their usability-focused research queries. While they primarily focus on enhancing microservices' usability, it's essential to consider that a user-friendly microservices architecture often begins with effective identification and design. The papers explore the learning barriers in end-user systems [32], the needs of developers for microservices deprecation [33], the failures of microservices documentation [34], what makes microservices difficult to learn [1], and microservices usability [35]. Understanding user perspectives is a crucial aspect of the broader microservices identification and development process.

Answers of RQ1: Empirical studies related to microservices identification often use diverse methods like large datasets, case studies, and user studies to reveal challenges and solutions. These typically focus on usability and maintainability.

3) TOOLS AND TECHNIQUE PROPOSALS

When it comes to migrating from a monolithic system to microservices, various tools and techniques are proposed to aid in microservices identification. These proposals aim to tackle existing issues in the field and offer potential solutions to migration-related challenges. Like the ''New Tools and Techniques'' category, these proposals address previously identified problems and suggest possible solutions. For instance, some propose automated techniques for identifying microservices based on their functionality or dependencies, while others suggest tools for visualizing the structure of a monolithic system and identifying potential microservices. However, it's important to note that these proposals are preliminary and lack comprehensive solution specifications or thorough evaluations.

Answers of RQ1: Tools and techniques proposals related to microservices identification typically seek to highlight existing concerns in the field, and provide potential approaches to resolving these problems.

4) SURVEYS

Like this research paper, surveys of existing literature aim to provide a rigorous evaluation of a research topic [12]. Typically, the surveys presented in this paper begin with a research topic and examine existing literature to provide a perspective on the subject at hand. Our dataset comprises five microservices identification-related surveys. Abdellatif et al. [36] reviewed 41 studies from 2004 to 2019, aiming to

identify inputs, processes, outputs, and the usability of service identification approaches for modernizing monolithic software. Their taxonomy covered broader contexts, assisting practitioners. Our focus, in contrast, is on identifying microservices when migrating from a monolithic to a microservices-based system. Ponce et al. [37] conducted a swift evaluation of the transition from monolithic to microservices architecture. They analyzed 20 research publications to investigate migration methods, their application to different systems, validation techniques, and encountered challenges. The study identified Model-Driven (MD), Static Analysis (SA), and Dynamic Analysis (DA) as migration approaches. Wolfart et al. [2] investigate the migration from monolithic systems to microservices, identifying 11 migration drivers like improved scalability, autonomous deployment, streamlined maintenance, team autonomy, and more. They also outline eight tasks across the initiation, planning, execution, and monitoring phases in the modernization process. The extensive review in [38] classifies research into distinct approach categories: Static Code Analysis (SCA), Meta-Data Assistance (MDA), Workload-Data Assistance (WDA), and Dynamic Microservices Composition (DMC). It evaluates these methods based on different criteria and offers a decision guide for them. However, it does not cover microservices decomposition techniques. This comprehensive review spans diverse topics, encompassing greenfield microservices development and monolithic application decomposition. Abgaz et al. [39] propose a framework for decomposing monoliths into microservices. Their findings reveal an early stage of progress in monolith decomposition, with a need for methods to integrate various data types, and insufficient tool support.

Answers of RQ1: Survey papers, like this systematic literature review, typically seek to present an overview of a subject using existing literature to provide clarity for their given subject and allow for effective stepping-stones for future research. The survey papers we reviewed delve into topics related to microservices identification evolution without a central focus on microservices identification evolution itself.

5) DATASETS

As a result of our investigation into microservices identification research, we discovered several articles focused on developing datasets related to microservices identification. For instance, Bandeira et al. [40] presented a dataset containing 1,043 microservice-related technical posts from StackOverflow, while Rahman et al. [41] described a dataset that includes web application microservices and their dependencies. Furthermore, Brogi et al. [42] set the groundwork for the first reference dataset of microservice-based applications. These datasets are crucial for the development and evaluation of precise and efficient microservices identification techniques.

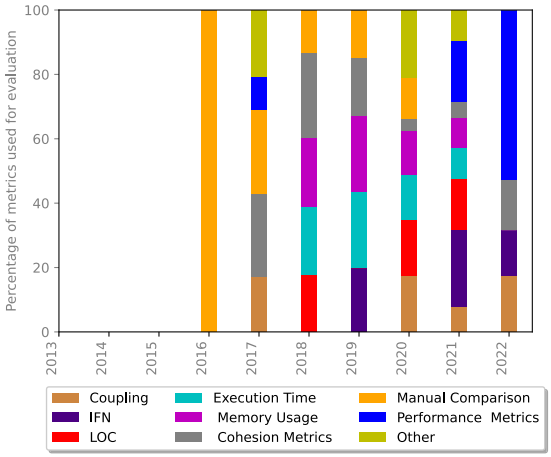

FIGURE 4. Keyword cloud of primary research.

B. MICROSERVICES IDENTIFICATION RESEARCH EVALUATION

We aim to ascertain how microservices identification research is typically evaluated when migrating from a monolithic to a microservices architecture. Identifying microservices requires typically more than merely observing a system manually. Studies rely on a variety of evaluation methods and software metrics to evaluate their findings.

We identify a variety of microservices identification evaluation techniques. Multiple subject systems were evaluated empirically utilising quantitative metrics such as coupling, Interface Number (IFN), Lack of Cohesion (LOC), and Structural Modularity (SMQ). Additionally, case studies were conducted using a singular subject system to acquire subject-related metrics and outcomes. Furthermore, user studies were conducted employing survey techniques and user or developer interviews. Focusing on the five paper categories, we identify the evaluation metrics employed in these papers. We uncovered 31 distinct evaluation metrics utilised in our publication sample. We grouped metrics that occurred fewer than five times and lacked known statistical properties into global metric types, such as qualitative metrics, and others. We obtained 9 metric categories as a result. Their annual tendencies are shown in Figure 4. Using the data we uncovered, we can see that while cohesion metrics are still widely used, performance metrics (e.g., precision, recall, etc.) are gaining significant popularity, particularly for assessing the performance of candidate microservices in cloud environments. However, a large proportion of papers continue to employ a wide range of non-standard absolute value metrics. CPU usage and network overhead are among the absolute value metrics used to evaluate experiments and instruments [43], [44]. None of these metrics are flawed, but the lack of standardisation makes it difficult to compare comparable experiments and evaluate progress.

1) NEW TOOLS AND TECHNIQUES

As discussed in Section IV-A, a significant portion of existing literature is dedicated to introducing new tools and techniques for microservices identification during migration from monolithic systems. Surprisingly, many of these tools and techniques lack a formal evaluation of their effectiveness. While the authors may have conducted evaluations, they are often not presented formally. For instance, some tools are introduced in brief papers and evaluated in subsequent publications, as seen in the case of [45]. Therefore, consumers should exercise caution when seeking an evaluation of a tool's reliability. Recent publications have shown a growing interest in using standardized metrics such as precision, recall, and F1-score as part of evidence for the effectiveness of supervised machine learning approaches [45], [46], [47], [48]. Over the past decade, there has been an increasing trend in the use of standardized metrics for experimentation, as depicted in Figure 4. In cases where it may be challenging to determine recall, as in the detection of changes in the utilization of a mined framework [49], authors often resort to manual comparisons or cohesion measures instead [50], [51].

2) EMPIRICAL STUDIES

All empirical studies focused on quantitative analysis to evaluate their findings regarding microservices identification during migration from monolithic to microservices-based systems. Depending on the study, metrics such as changes in microservices (e.g., addition, modification, removal) [52], changes in lines of code [53], code smells [54], and microservices' popularity [55] were evaluated. The most common evaluation criteria for identifying microservices included the analysis of the business domain and system characteristics, such as functional requirements, domain models, and transactions. Quantifying microservice changes through added/modified/removed microservices appeared to be a common practice in case studies on microservices identification.

While the majority of microservices case studies examined and quantified microservice changes, some also relied on qualitative assessments [26], [56]. Qualitative data like this requires manual extraction by the authors. Case studies are well-suited for uncovering new evaluation metrics for microservices to reveal previously unknown information, such as the types of ripple effects caused by changes to software ecosystems, and microservices migration issues [57]. Therefore, case studies may introduce new, relatively rare metrics as they seek to identify previously undiscovered factors. The insights gained from case studies can subsequently be applied to larger-scale empirical studies of various microservices.

3) TOOLS AND TECHNIQUE PROPOSALS

Migration from a monolithic system to a microservices architecture presents numerous challenges and complexities, which are often discussed in presentations and expert panels focusing on microservices identification. These presentations delve into the characteristics of the business domain and the system. However, many of these papers lack clear and transparent evaluation criteria for their methods. Some recommended practices papers propose specific software metrics that could benefit from developer expertise, such as metrics related to coupling reduction [58]. Additionally, tool proposals often include evaluation metrics to assess the tool's precision and effectiveness, alongside user studies designed to collect feedback from developers [26], [56].

4) SURVEYS

During the migration from a monolithic system to microservices, there are two main categories of survey papers. The first category focuses on existing literature, for example the survey conducted by [39]. Surveys of this type examine and summarize existing techniques but often do not employ specific metrics to evaluate the papers included in their findings. Instead, they rely on the evaluations presented within each surveyed paper. Additionally, surveys of this type typically define a specific scope and set of criteria, which are manually assessed by the author. Similarly, in our systematic literature review, we rely on the evaluations provided in the papers we sampled. However, we also incorporate quantitative data to identify publishing and evaluation trends, as well as the emergence of microservices identification sub-fields.

The second category of survey papers provides results obtained from queries used to gather information from participants. These papers offer quantifiable data that can be analyzed in various ways. For example, [59] includes raw data from survey responses conducted in a related study, along with a quantitative evaluation of those responses. Meanwhile, other works, like the study by [60], investigate the behavior of programmers in relation to specific microservices identification-related tasks. It is important to note that there is no standardized dataset or evaluation method commonly used for microservices identification surveys. Current evaluation methods tend to be tailored to specific papers. The absence of a standardized evaluation methodology within the community is a matter that should be addressed, as it hinders research comparisons and makes it challenging to determine the progress made in microservices identification.

5) DATASETS

The shortage of datasets is a significant challenge, as most studies resort to utilizing open-source applications of relatively modest size, typically with fewer than 200,000 lines of code. This shortage of available datasets and benchmarking data represents a relatively underexplored area in microservices identification research. Researchers typically opt for monolithic applications for their experiments due to the ready availability of monolithic source code, often sourced from Open Source Software. However, there has been a recent

emergence of monolithic applications that come paired with their corresponding microservices implementations. Some noteworthy examples include JPetStore [22], [34], [61], [62], DayTrader [23], [61], [63], Acme Air [61], [63], Petclinic [8], [63], [64], and Cargo Tracking System [21], [65]. Other studies featured in this review employ distinct codebases to illustrate and validate their proposed methods.

Answers of RQ1: Empirical evaluation in microservices identification studies in general has not yet converged on specific styles and metrics. A surprising number of microservices identification tools and techniques do not include any empirical evaluation, while studies with similar tools and techniques assess precision metrics and IFN. Meanwhile, empirical studies on microservices identification rely on various metrics, with the frequency of cohesion metrics being the most common, but not always. Survey papers, tools, and techniques proposal papers similarly present a variety of evaluation criteria with no clear standards. While some flexibility is indeed required to accommodate various research goals, there is still work to be done to evaluate similar research goals using consistent evaluation styles and metrics.

V. SEMINAL AND RECENT PUBLICATIONS

To answer RQ2: What is the current state of the art in microservices identification research ?, we first present publication trends within the state of the art. We then concentrate on the seminal and most recent concepts and research works. We chose these seminal works based on the novelty of their content, and the number of works that present similar ideas and build on these seminal works. This section is also divided by publication contribution category, as in Section IV-A.

A. NEW TOOLS AND TECHNIQUES

A variety of tools and techniques have been developed over time to facilitate microservices identification and migration from monolithic systems. These tools and techniques are generally focused on decoupling monolithic systems into microservices while ensuring their functionality and efficacy. We categorize microservices identification tools and techniques into general topics such as documentation, examples, migrations, and recommendations, and others. As stated in Section III-D, these tool topics were either identified in prior surveys [2], [36], [37], [38], [39], or by using publication keywords, titles, abstracts as well as our own judgement. We provide a general overview of the state-of-the-art for each tool topic.

- *Microservices Documentation Tools:* State-of-the-art documentation tools and techniques include using Stack Overflow posts to supplement documentation for lexical queries [40], dynamically generating microservice documentation based on service interactions [44], and employing Natural Language Processing (NLP) techniques to extract and summarize microservice documentation [7], [66]. Other tools focus on visualizing microservice dependencies [67] or extracting the communication patterns between microservices [32], [68] to aid in documentation efforts.
- *Microservices Examples Tools:* The examples aid in comprehending how microservices operate in the real world and facilitate their incorporation into the larger system. This will enable developers to comprehend how microservices interact and how to utilise them effectively. Exemplary microservices tools have been lauded for their utility in comprehending how microservices function [62], [69]. *MSExtracter* [20] and *Decomposer* [50] extract microservice examples from existing source code. In order to identify instances of microservices, techniques employing log visualisation [70] have also been utilised.
- *Microservices Migration Tools:* Existing services must be identified and partitioned in order to migrate from a monolithic system to microservices. This task has been facilitated by the creation of tools and methods. *Microservice Miner* [69] employs source code analysis to determine service boundaries and interdependencies. *Microservice Miner* [71] provides a model-driven migration strategy from monolithic to microservices. Other tools, such as the *Microservice Migration Assistant (MMA)* [72], use static and dynamic analysis to identify microservice entry points and their interactions. *ExploreViz* [73] provides a graphical user interface for visualising the structure of a monolithic system and investigating potential microservices.
- *Microservices Recommendation Tools:* Recommendations are exemplary design, implementation, and maintenance practises for microservices. Recommendations may consist of coding standards, security policies, and release management procedures [74]. Popular recommendation tools for microservices include Microservices.io [75] and Netflix OSS [76], which offer templates, patterns, and guidance for creating microservices. These tools can help developers save time and effort in identifying and separating existing services, leading to a smoother and more efficient migration from a monolithic system to microservices. Various industrial solutions have emerged to tackle the challenge of breaking down monolithic systems into microservices. Notable examples include IBM's *Mono2Micro tool* [2] and Amazon's *AWS Microservice Extractor for NET*.[3]
- *Microservices Usage Mining Tools:* Static and dynamic data extraction techniques have been proposed

[2]https://www.ibm.com/cloud/blog/announcements/ibm-mono2micro
[3]https://aws.amazon.com/about-aws/whats-new/2021/11/aws-microservice-extractor-net/

to identify microservices. *Kieker* [22], [34] gathers dynamic information from Java applications [77], while *Dbeaver* [73] is employed to retrieve runtime database table access within a monolithic system. These techniques involve analyzing the codebase to identify modules with high cohesion and low coupling, which are good candidates for microservices [78]. In addition, tools such as *Scipy Python Library* [52], *Arcan* [79], and *DISCO* [80] can be used to visualise and analyse the communication patterns between microservices, thereby facilitating their identification and separation from a monolithic system.

Different tools are vital for transitioning from a monolithic system to microservices. They offer a wide range of solutions, aiding developers in achieving a successful transition.

Answers of RQ2: State-of-the-art tools and techniques related to microservices identification, aim to, in order of importance, enhance microservices utilization, facilitate adaptation to changes, automate microservices migration, offer microservices recommendations, mitigate microservices misuse, and enhance microservices documentation and examples.

B. EMPIRICAL STUDIES

In the context of migrating from a monolithic system to microservices, recent empirical research extensively investigates the identification and management of microservices.

- *Microservice Identification:* These studies delve into various aspects of identifying microservices, addressing concerns like decomposing the monolithic system, defining service boundaries, understanding granularity, and managing dependencies. A systematic approach to microservice identification can reduce the risk of dependency failures and improve maintainability [81]. Automated tools are suggested to assist developers in identifying service boundaries [82]. Recent studies propose multiple techniques, including clustering-based, dependency-based, and data-driven approaches, for microservice identification [83], [84].
- *Microservice Maintenance:* Once microservices are identified and deployed, ensuring their long-term maintainability becomes crucial. This encompasses testing, monitoring, debugging, versioning, and evolution [85], [86]. Effective diagnostic techniques are essential for problem resolution, and versioning allows governance over microservices' evolution [87]. Some studies explore the application of heuristics for tasks like defect diagnosis and performance optimization [88].

Recent empirical studies offer valuable insights into addressing the challenges and best practices for microservice identification and management, which are vital for a successful transition from a monolithic system to microservices.

Answers of RQ2: Empirical studies on microservices identification maintainability typically focus on challenges related to the rate of microservice changes, and the impact of changes.

C. TOOLS AND TECHNIQUE PROPOSALS

The proposals for tools and techniques in this context primarily address existing issues while suggesting solutions for future research. They aim to advance microservices identification and maintenance. Recent proposals highlight the importance of distinguishing between different microservices [89] and developing techniques for extracting microservices from monolithic codebases [90]. Another proposal focuses on challenges such as establishing precise connections between microservices, capturing application context and synthesizing documentation [19]. Additionally, techniques employing genetic algorithms [91], and natural language processing for microservices identification requirements gain popularity [7]. These proposals provide valuable insights into the evolving needs of researchers and developers in microservices identification and maintenance.

Answers of RQ2: Tools and technique proposals aim to enhance microservices identification and maintenance by addressing issues such as distinguishing microservices and employing genetic algorithms and natural language processing.

D. SURVEYS

Surveys inherently highlight pioneering concepts and the latest advancements. As mentioned in Section IV-A, we've identified five survey papers in line with the methodology outlined in Section III. These papers explore various aspects of microservices, including recommendation systems [92], [93], software ecosystems [94], property inference techniques [63], and fusion techniques [3]. We leverage metrics, classifications, and challenges drawn from previous surveys [8], [93], [95], [96], [97] to reinforce our findings and categorize the tools and techniques discussed in microservices identification and empirical studies in Sections V and VI. Additionally, these survey papers pinpoint ongoing challenges and future research directions within their domains. While some of these issues have been addressed since the surveys were conducted, a few persist, and we revisit them in Section VI, alongside our own discoveries.

Answers of RQ2: Surveys associated to microservices identification tend to highlight the state-of-the-art in research as well as current research challenges and future research directions.

E. DATASETS

The focus of papers primarily centred on datasets is future research. Two papers that introduce datasets are considered primary contributions [41], [42]. However, publications categorized under different primary contributions (e.g., Empirical studies) may also include datasets as secondary contributions. For instance, some publications contribute methodologies [98], [99] alongside datasets. The practice of open-sourcing research datasets is emerging as a research area that needs to be addressed.

Answers of RQ2: State-of-the-art datasets are essential. The emergence of open-sourcing research datasets is an area requiring attention.

Answers of RQ2: We discussed influential and recent works in microservices identification. Their objectives include simplifying microservices identification, handling microservices changes, and offering recommendations. They also seek to minimize microservices misuse and enhance microservices documentation. They propose that future efforts should concentrate on developing automated microservices identification tools and creating datasets for consistent migration evaluations.

VI. CURRENT AND FUTURE CHALLENGES

To address RQ3: What are the current and future challenges in microservices identification during the transition from a monolithic system? We manually identified both existing and unsolved challenges, as presented in Table 3. Despite the rapid growth in microservices research and the emergence of promising tools, significant challenges persist in the field of microservices identification.

Challenges related to microservices identification are scattered throughout the literature, often intertwined with both advancements and persistent obstacles. While conducting this literature review, we compiled a list of challenges mentioned in published works. Challenges for which solutions have been proposed are considered *existing challenges (EC)*, while those without known solutions are regarded as emerging or *unsolved challenges (UC)*. We have supplemented these unsolved challenges with additional insights from our review.

We identified existing challenges in the field of new tools and techniques, as well as empirical studies. No existing challenges were found in proposals or surveys, only unsolved ones. Among these challenges, we believe that Lehman's eighth law, the Feedback System [100], represents a significant barrier to future research in microservices identification.

A. NEW TOOLS AND TECHNIQUES

Existing Challenges: **Issue**: Limited research has focused on microservices identification tools designed specifically for Web APIs, highlighting the need for more comprehensive studies and tools that address the unique challenges posed by Web APIs in microservices identification (EC-10). Identifying microservices involves considering factors such as service boundaries, service quality, and the absence of exhaustive microservices listings [52], [98]. **Propositions**: Researchers can leverage existing research, such as microservices migration approaches [47], [50], [64], [72], [107], [108], high-quality code summary generation [109], misuse identification [99], and the use of relational topic models for examples [62], as foundations for enhancing microservices identification tools (EC-11). Modern migration techniques should explore hybrid approaches (EC-12) that combine API-side learning with client-side learning [105] and domain adaptation techniques (EC-13) to address out-of-vocabulary problems, a current challenge in microservices identification [106]. **Issue**: The development of identification, recommendation, and misuse detectors for microservices is an ongoing challenge. **Propositions**: Addressing these challenges requires active involvement of microservices users, as they are the ones most affected. Furthermore, tools designed to assist with these issues should offer support for additional programming languages and Web APIs.

Unresolved Challenges: **Issue**: While numerous tools and techniques have been developed to address microservices identification issues, most tools focus on specific challenges and do not fully account for feedback cycles involved in microservices identification. Although individual tools demonstrate promising results [8], [101], [110], none can claim to be 100 % effective in resolving their target problem. With the emergence of machine learning approaches as potential solutions to key microservices identification issues [3], [23], [47], [88], questions arise regarding the suitability of current approaches for user adoption, their applicability to all issues, and the need for performance enhancements before widespread tool adoption (UC-3). Fuzzy and ambiguous intent (UC-4) and the rapid evolution of software services using microservices, such as IoT devices, present challenges in the evolution of microservices [10], [107]. **Propositions**: Effective microservices engineering should aim to resolve technical issues stemming from microservices and bridge the knowledge gap between microservices developers and users (UC-5). New tools are needed to help microservices developers produce user-friendly microservices [8], [101], [110] (UC-6), and improved techniques should assist microservices users in understanding how to use these microservices [2] (UC-7). Researchers should seek to understand what constitutes a ''good'' microservice and why users prefer one microservice over another to address these challenges.

Issue: Many organizations aspire to migrate from monolithic to microservices architecture [47], [50]. Identifying the appropriate microservices to separate from the monolithic system remains a challenging task. The effectiveness of existing monolithic systems' decomposition into microservices is still uncertain (UC-8).

TABLE 3. Open challenges in microservices identification

Challenge types	Paper types	Challenges
Existing challenges	New tools and techniques	*EC-1* Need for accurate microservice identification tools combining textual, syntactic, and semantic techniques [7], [21]
		EC-2 Lack of commercially viable microservice identification solutions [76], [103]
		EC-3 Need to integrate domain-specific information when identifying microservices [8]
		EC-4 Employing systematic evaluation methods in empirical evaluations [104]
		EC-5 Producing more theories regarding software ecosystems and microservices [105]
		EC-6 Difficulty in studying and identifying microservices within software ecosystems [105]
		EC-7 Addressing difficulty in automating repetitive software changes when migrating to microservices [106]
		EC-8 Taking up the problem of integrating testing with code recommendations for microservice migrations [5]
		EC-9 Heavy reliance on human intervention to verify microservice recommendations [7]
		EC-10 Limited research on microservice identification tools for web APIs [52]
		EC-11 Difficulty identifying appropriate microservices when migrating from a monolithic system [52]
		EC-12 Need for hybrid approaches combining API and client-side learning for microservices [107]
		EC-13 Domain adaptation techniques to address out-of-vocabulary problems [108]
	Empirical studies	*EC-14* Defining best-fit microservices [106]
		EC-15 Automatically identifying factors driving microservices changes [31]
		EC-16 Need for research into program semantics and dependencies [101]
		EC-17 Need for tools to deploy problem solutions to multiple microservices simultaneously [88]
Unsolved Challenges	New tools and techniques	*UC-1* Need for accurate microservice identification tools using multiple techniques
		UC-2 Impact of context sensitivity on microservice identification and migration process
		UC-3 Unclear effectiveness of machine learning approaches for microservices identification
		UC-4 Fuzzy and ambiguous intent
		UC-5 Reducing knowledge gap between microservices developers and users
		UC-6 Need for tools to simplify microservices development
		UC-7 Need for improved techniques for microservices users to understand usage
		UC-8 Effectiveness of decomposing monolithic systems into microservices
		UC-9 Efficient sharing of microservice identification burden between application and infrastructure developers
		UC-10 Establishing a feedback cycle for microservices developers to improve their services
	Empirical studies	*UC-11* Lack of systematic evaluation methodology for identifying microservices
		UC-12 Determine if microservices identification issues are language-specific or common to all programming languages
		UC-13 Need for future research to generalize to languages other than Java
		UC-14 Lack of standardized method for comparing results across microservices studies
	Datasets	*UC-15* Need for more datasets for research and comparison of microservices identification techniques

Issue: Microservices identification has garnered significant attention in migration research but remains unresolved. The assumption underlying most current approaches [20], [23], [38] is that microservices identification is solely the responsibility of application developers. **Propositions**: Research should explore the potential efficiency gains of shifting some of the burden to infrastructure developers (UC-9), such as having them provide tools or scripts for microservices identification. Additionally, tools should be developed to streamline microservices engineering and reduce the identification workload on the application side.

Issue: Several tools have been developed to analyze monolithic systems and identify potential microservices [8]. **Propositions**: This information should be utilized to establish a feedback cycle to assist application developers in enhancing their microservices (e.g., using microservice dependencies as areas for improvement [21]) (UC-10). Over the past decade, migration research has predominantly focused on application developers rather than infrastructure developers.

B. EMPIRICAL STUDIES

Existing Challenges: **Issue/Proposition**: Studies have revealed the need for future work on microservices developers and microservices development to support the migration of monolithic systems to microservices [57], define best fit microservices [104] (EC-14), and automatically identify factors driving microservices changes [31] (EC-15). **Issue/Proposition**: In their study on microservices identification, [7] emphasise the need for future research into the semantics and dependencies of programs (EC-16), as well as the need for tools that can manage alternative patterns for the same microservice. **Issue/Proposition**: Aksakalli et al. [87] have proposed the need for tools to deploy problem solutions to multiple microservices simultaneously (EC-17).

Unresolved Challenges: **Issue**: The majority (96%) of empirical studies on microservices identification focus on systems written in the Java programming language. A small percentage (≤ 5 % each) of empirical investigations cover other languages such as C, C++, COBOL, and Python. **Proposition**: Future research should be extended to languages other than Java (UC-13). **Issue**: A large proportion (74%) of empirical studies do not use statistical analyses to evaluate their findings. The majority of these studies exhibit metrics such as Lines-Of-Code (LOC) or the number of service changes; however, there is currently no method to normalise these results so that they can be compared across studies (UC-14). **Proposition**: Comparison of the migration methods, particularly across programming languages, remains a challenge.

C. DATASETS

Unsolved Challenges: **Issue**: The lack of widely accepted and up-to-date datasets makes it difficult to evaluate and compare various microservices identification techniques. **Proposition**: To advance research in microservices identification and enable direct comparisons of different techniques (UC-15), there is a pressing need for more datasets. However, producing such datasets is challenging due to the subjective and context-sensitive nature of microservices identification.

D. OTHERS

Other research objectives on identifying microservices, tools and technique proposals, and surveys are scarcer, so we discuss them together in this section.

Existing Challenges: **Issue/Proposition**: One of the challenges of migrating from a monolithic system to microservices is determining which microservices to employ. There is a need for tools that can accurately identify microservices by combining textual, syntactic, and semantic techniques (EC-1). Although some programmes, such as MOGA-WSI [101] and MicroserviceExtraction [8], have attempted this, a commercially viable solution has not yet been developed (EC-2). When identifying microservices, it is also essential to integrate domain-specific information, which has been attempted with varying degrees of success (EC-3). However, it appears that current solutions are context-dependent, and more research is required in this area.

Issue/Proposition: While there have been some studies attempting to develop theories about microservices [25], [35], [60], [92], the majority of tools and research appear to be closely tied to factors such as microservices ecosystems and the programming languages used for microservices (UC-11). We have found that there is currently no established systematic methodology for evaluating the identification of microservices. Although their survey laid the groundwork for comparing microservice identification techniques, there has been limited progress in implementing a systematic evaluation methodology (EC-4). The reasons for this lack of adoption remain unclear but could be attributed to limited exposure or the inherent complexities associated with the proposed approach. Addressing this challenge should be a priority to enhance the visibility of existing methodologies and guide future research towards more systematic and comparable evaluations.

Issue/Proposition: According to [103], theories regarding software ecosystems and the services they entail are frequently either too general (EC-5) or too abstract. Due to the high variability of the field, it is difficult to study software ecosystems, and the same difficulty applies to identifying microservices within these ecosystems (EC-6).

Issue/Proposition: In [104], the authors highlight several open challenges with respect to automating repetitive software changes when migrating to microservices. One of the challenges is finding input examples to automate the process (EC-7). Integrating testing with code recommendation and dealing with various levels of code granularity for microservice recommendations and migrations also remain open challenges (EC-8). Current recommendation tools rely heavily on human intervention to determine the correctness of the recommendation (EC-9). Although [104] have attempted to automate the process of identifying microservices, this

challenge has not been fully solved. More work is needed to extract code examples relevant to user queries and determine the similarity of multiple examples.

Unsolved Challenges: **Issue**: Currently, there is a lack of standardized benchmarks to evaluate the performance of microservice identification techniques. The results of these techniques are therefore at the mercy of the dataset and evaluation methodologies chosen by their authors, preventing comparisons between techniques. **Proposition**: Future research should seek to use a standard evaluation such as the one provided by [102] to improve the ease of comparison between various approaches (UC-1).

Issue: Identifying microservices is a context-sensitive problem, and there is a need to incorporate domain-specific information into tools to account for this context sensitivity. However, it is unclear how to best support the context-sensitive nature of microservice identification tools and how their usage might affect the overall migration process (UC-2).

Issue: Few studies have attempted to determine whether the severity of the various problems in identifying microservices is present in all programming languages (UC-12). **Proposition**: Systematic studies to determine the impact of identifying microservices during migration and the helpfulness of microservice identification tools are required to understand whether such aid is universally required or language dependent.

Answers of RQ3: Table 3 [4] *summarises and labels existing challenges (EC-1 through EC-17) and unsolved challenges (UC-1 through UC-15) identified during this systematic literature review. It shows that existing and unsolved challenges concern new tools and techniques and empirical studies first. We also consider unsolved challenges with datasets. They are concerned first and foremost with microservices identification during migration from a monolithic system to microservices, and the evaluation/validation of microservices identification tools and their results.*

VII. THREATS TO VALIDITY

Construct validity. While we acknowledge that the search phrase "Microservices Identification" may not be perfect, and different search queries could yield additional results, we have included a substantial number of studies to provide a comprehensive representation of the field. Our taxonomy was developed with some ad-hoc elements, introducing potential subjectivity bias [111]. To address this, we employed classifications found in existing papers, synonyms for established terminology.

[4]It shows the main references presenting existing challenges. Emerging unsolved challenges are indirectly referenced because they are recently emerging and have not yet been thoroughly discussed and addressed in the literature.

External validity. Although it is improbable that we have identified every paper related to microservices identification, we believe that our selection of publications is indicative of the state of the art in this field. We are confident that the majority of relevant works are included, and the trends and findings presented reflect the current state-of-the-art. Our efforts to mitigate this included utilizing six different publication search engines and implementing forward and backward snowballing to capture papers that might have been missed.

Internal validity. To minimize potential biases, we, drawing on our expertise in microservices identification, agreed on the selection criteria and paper categorization. The categories used for classification were also collectively determined. While the majority of paper selection and classification was conducted by one author, we conducted a test-retest reliability assessment to ensure internal consistency, yielding excellent results.

VIII. CONCLUSION

In this systematic survey of the literature on microservices identification, we uncovered the publication trends as well as questions and goals common in the literature. We answered three research questions: RQ1: How has microservice identification research evolved? RQ2: What is the current state of microservice identification research? RQ3: What are the current and future challenges in microservice identification?

We observed that there are five research goals, in Section IV-A: *new tools and techniques, empirical investigations, proposed tools, surveys, and datasets*. In Section IV-B, we observed a variety of evaluation metrics, with cohesion metrics, IFN, and LOC being the most common, but these metrics aren't consistently applied in all studies. Consequently, it's challenging to compare the effectiveness of methods proposed in different studies. Future research should aim to establish a standard set of metrics for monolith analysis and microservices identification. Metrics are not only essential in the analysis and identification phases; there's also a notable lack of consistent evaluation of the resulting microservices, which calls for the publication of datasets. These datasets should encompass key elements, such as the monolith source code, the extracted microservices, and the metrics used at different stages. **To facilitate systematic comparisons and advancements, we recommend adopting standard benchmarks and evaluation techniques. Furthermore, exploring the impact of microservice migration and assessing their evolution represents valuable research directions.**

We studied the tools and techniques from existing literature and found that their primary purposes are to improve microservices identification, offer guidance for microservices recommendations, assist in microservices migration, mitigate issues in microservices usage, and enhance microservices documentation. **We recommend enhancing these tools with domain-specific knowledge, developing tools designed for identifying microservices within**

Web APIs, and developing tools to help microservices developers improve their microservices. We also suggest exploring the use of NLP techniques to address challenges in microservices identification.

Empirical research on microservice identification predominantly focuses on usability and maintainability. It delves into disruptive changes, integration challenges, standards, usage, abuse, and documentation. **We recommend studying the influence of microservices on application scalability.**

A lot of the research done so far has concentrated on a limited set of programming languages, with a strong preference for Java. However, many big business monolithic systems have been constructed using languages like COBOL and C/C++. **We believe it's crucial to give more consideration to this aspect. We propose exploring various programming languages beyond Java to enhance adaptability and uncover shared or distinct factors.**

The need for faster software updates and the rise of cloud computing for distributed software systems make breaking down monolithic systems into microservices important for software development. We encourage researchers to make their benchmarks and datasets publicly available. While challenges like ongoing change and complexity persist, the next frontier lies in mastering feedback systems in microservice identification. We believe this study will benefit existing work and inspire future research in microservice identification.

REFERENCES

[1] S. Newman, *Building Microservices*. Sebastopol, CA, USA: O'Reilly Media, 2021.

[2] D. Wolfart, W. K. G. Assunção, I. F. da Silva, D. C. P. Domingos, E. Schmeing, G. L. D. Villaca, and D. D. N. Paza, "Modernizing legacy systems with microservices: A roadmap," in *Proc. Eval. Assessment Softw. Eng.* New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 149–159, doi: 10.1145/3463274.3463334.

[3] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn, "Microservices migration patterns," *Software: Pract. Exper.*, vol. 48, no. 11, pp. 2019–2042, Nov. 2018, doi: 10.1002/spe.2608.

[4] G. Kecskemeti, A. C. Marosi, and A. Kertesz, "The ENTICE approach to decompose monolithic services into microservices," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2016, pp. 591–596, doi: 10.1109/HPCSim.2016.7568389.

[5] A. Selmadji, A.-D. Seriai, H. L. Bouziane, C. Dony, and R. O. Mahamane, "Re-architecting OO software into microservices," in *Proc. Service-Oriented Cloud Comput.*, 2018, pp. 65–73, doi: 10.1007/978-3-319-99819-0_5.

[6] A. Levcovitz, R. Terra, and M. Tulio Valente, "Towards a technique for extracting microservices from monolithic enterprise systems," 2016, *arXiv:1605.03175*.

[7] S.-P. Ma, Y. Chuang, C.-W. Lan, H.-M. Chen, C.-Y. Huang, and C.-Y. Li, "Scenario-based microservice retrieval using Word2 Vec," in *Proc. IEEE 15th Int. Conf. e-Business Eng. (ICEBE)*, Oct. 2018, pp. 239–244, doi: 10.1109/ICEBE.2018.00046.

[8] G. Mazlami, J. Cito, and P. Leitner, "Extraction of microservices from monolithic software architectures," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 524–531, doi: 10.1109/ICWS.2017.61.

[9] I. Oumoussa, S. Faieq, and R. Saidi, "Microservices: Investigating underpinnings," in *Proc. Int. Conf. Netw., Intell. Syst. Secur.*, 2022, pp. 343–351, doi: 10.1007/978-3-031-15191-0_33.

[10] D. Lu, D. Huang, A. Walenstein, and D. Medhi, "A secure microservice framework for IoT," in *Proc. IEEE Symp. Service-Oriented Syst. Eng. (SOSE)*, Apr. 2017, pp. 9–18, doi: 10.1109/SOSE.2017.27.

[11] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of microservices-enabled fog applications," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 22, p. e4436, Nov. 2019, doi: 10.1002/cpe.4436.

[12] B. Kitchenham and S. Charterss, "Guidelines for performing systematic literature reviews in software engineering," School Comput. Sci. Math., Softw. Eng. Group, Keele Univ., Keele, U.K., Tech. Rep. EBSE-2007-01, 2007.

[13] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015, doi: 10.1016/j.infsof.2015.03.007.

[14] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*. Hoboken, NJ, USA: Wiley, 2008, doi: 10.1002/9780470754887.

[15] Z. M. Jiang and A. E. Hassan, "A survey on load testing of large-scale software systems," *IEEE Trans. Softw. Eng.*, vol. 41, no. 11, pp. 1091–1118, Nov. 2015.

[16] R. Huang, W. Sun, Y. Xu, H. Chen, D. Towey, and X. Xia, "A survey on adaptive random testing," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2052–2083, Oct. 2021, doi: 10.1109/TSE.2019.2942921. https://doi.org/10.1109/TSE.2019.2942921

[17] M. Kuhrmann, D. M. Fernández, and M. Daneva, "On the pragmatic design of literature studies in software engineering: An experience-based guideline," *Empirical Softw. Eng.*, vol. 22, no. 6, pp. 2852–2891, Dec. 2017, doi: 10.1007/s10664-016-9492-y.

[18] O. Al-Debagy and P. Martinek, "A microservice decomposition method through using distributed representation of source code," *Scalable Comput., Pract. Exper.*, vol. 22, no. 1, pp. 39–52, Feb. 2021, doi: 10.12694/scpe.v22i1.1836.

[19] A. Furda, C. Fidge, O. Zimmermann, W. Kelly, and A. Barros, "Migrating enterprise legacy source code to microservices: On multitenancy, statefulness, and data consistency," *IEEE Softw.*, vol. 35, no. 3, pp. 63–72, May 2018, doi: 10.1109/MS.2017.440134612.

[20] I. Saidani, A. Ouni, M. W. Mkaouer, and A. Saied, "Towards automated microservices extraction using muti-objective evolutionary search," in *Proc. Service-Oriented Comput., 17th Int. Conf.*, Toulouse, France, 2019, pp. 58–63, doi: 10.1007/978-3-030-33702-5_5.

[21] M. Daoud, A. El Mezouari, N. Faci, D. Benslimane, Z. Maamar, and A. El Fazziki, "A multi-model based microservices identification approach," *J. Syst. Archit.*, vol. 118, Sep. 2021, Art. no. 102200, doi: 10.1016/j.sysarc.2021.102200.

[22] W. Jin, T. Liu, Y. Cai, R. Kazman, R. Mo, and Q. Zheng, "Service candidate identification from monolithic systems based on execution traces," *IEEE Trans. Softw. Eng.*, vol. 47, no. 5, pp. 987–1007, May 2021, doi: 10.1109/TSE.2019.2910531.

[23] U. Desai, S. Bandyopadhyay, and S. Tamilselvam, "Graph neural network to dilute outliers for refactoring monolith application," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 1, pp. 72–80.

[24] T. Kinoshita and H. Kanuka, "Automated microservice decomposition method as multi-objective optimization," in *Proc. IEEE 19th Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2022, pp. 112–115, doi: 10.1109/ICSA-C54293.2022.00028.

[25] S. S. De Toledo, A. Martini, P. H. Nguyen, and D. I. K. Sjøberg, "Accumulation and prioritization of architectural debt in three companies migrating to microservices," *IEEE Access*, vol. 10, pp. 37422–37445, 2022, doi: 10.1109/ACCESS.2022.3158648. https://doi.org/10.1109/ACCESS.2022.3158648

[26] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices migration in industry: Intentions, strategies, and challenges," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2019, pp. 481–490, doi: 10.1109/ICSME.2019.00081.

[27] S. Eski and F. Buzluca, "An automatic extraction approach: Transition to microservices architecture from monolithic application," in *Proc. 19th Int. Conf. Agile Softw. Development: Companion*. New York, NY, USA: Association for Computing Machinery, May 2018, pp. 1–6, doi: 10.1145/3234152.3234195.

[28] J. Ghofrani and A. Bozorgmehr, "Migration to microservices: Barriers and solutions," in *Applied Informatics*. New York, NY, USA: Springer, 2019, pp. 269–281, doi 10.1007/978-3-030-32475-9_20.

[29] M.-D. Cojocaru, A. Uta, and A.-M. Oprescu, "Attributes assessing the quality of microservices automatically decomposed from monolithic applications," in *Proc. 18th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jun. 2019, pp. 84–93, doi: 10.1109/ISPDC.2019.00021.

[30] V. Lenarduzzi, F. Lomio, N. Saarimäki, and D. Taibi, "Does migrating a monolithic system to microservices decrease the technical debt?" *J. Syst. Softw.*, vol. 169, Nov. 2020, Art. no. 110710, doi: 10.1016/j.jss.2020.110710.

[31] A. V. Zarras, P. Vassiliadis, and I. Dinos, "Keep calm and wait for the spike! insights on the evolution of Amazon services," in *Proc. Adv. Inf. Syst. Eng.*, 2016, pp. 444–458, doi: 10.1007/978-3-319-39696-5_27.

[32] G. Vale, F. F. Correia, E. M. Guerra, T. de Oliveira Rosa, J. Fritzsch, and J. Bogner, "Designing microservice systems using patterns: An empirical study on quality trade-offs," in *Proc. IEEE 19th Int. Conf. Softw. Archit. (ICSA)*, Mar. 2022, pp. 69–79, doi: 10.1109/ICSA53651.2022.00015.

[33] J. Lewis and M. Fowler, "Microservices: A definition of this new architectural term," *MartinFowler. Com*, vol. 25, nos. 14–26, p. 12, 2014.

[34] Y. Zhang, B. Liu, L. Dai, K. Chen, and X. Cao, "Automated microservice identification in legacy systems with functional and non-functional metrics," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Mar. 2020, pp. 135–145, doi: 10.1109/ICSA47634.2020.00021.

[35] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-29044-2.

[36] M. Abdellatif, A. Shatnawi, H. Mili, N. Moha, G. E. Boussaidi, G. Hecht, J. Privat, and Y.-G. Guéhéneuc, "A taxonomy of service identification approaches for legacy software systems modernization," *J. Syst. Softw.*, vol. 173, Mar. 2021, Art. no. 110868, doi: 10.1016/j.jss.2020.110868.

[37] F. Ponce, G. Márquez, and H. Astudillo, "Migrating from monolithic architecture to microservices: A rapid review," in *Proc. Int. Conf. Chilean Comput. Sci. Soc. (SCCC)*, 2019, pp. 1–7, doi: 10.1109/SCCC49216.2019.8966423.

[38] J. Fritzsch, J. Bogner, A. Zimmermann, and S. Wagner, "From monolith to microservices: A classification of refactoring approaches," in *Software Engineering Aspects of Continuous Development and*. New York, NY, USA: Springer, 2019, pp. 128–141, doi: 10.1007/978-3-030-06019-0_10.

[39] Y. Abgaz, A. McCarren, P. Elger, D. Solan, N. Lapuz, M. Bivol, G. Jackson, M. Yilmaz, J. Buckley, and P. Clarke, "Decomposition of monolith applications into microservices architectures: A systematic review," *IEEE Trans. Softw. Eng.*, vol. 49, no. 8, pp. 4213–4242, Jul. 2023, doi: 10.1109/TSE.2023.3287297.

[40] A. Bandeira, C. A. Medeiros, M. Paixao, and P. H. Maia, "We need to talk about microservices: An analysis from the discussions on StackOverflow," in *Proc. IEEE/ACM 16th Int. Conf. Mining Softw. Repositories (MSR)*, May 2019, pp. 255–259, doi: 10.1109/MSR.2019.00051.

[41] M. Imranur, Rahman, P. Sebastiano, and T. Davide, "A curated dataset of microservices-based systems," 2019, *arXiv:1909.03249*.

[42] A. Brogi, A. Canciani, D. Neri, L. Rinaldi, and J. Soldani, "Towards a reference dataset of microservice-based applications," in *Proc. Softw. Eng. Formal Methods*, 2018, pp. 219–229, doi: 10.1007/978-3-319-74781-1_16.

[43] W. K. G. Assunção, T. E. Colanzi, L. Carvalho, J. A. Pereira, A. Garcia, M. J. de Lima, and C. Lucena, "A multi-criteria strategy for redesigning legacy features as microservices: An industrial case study," in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, 2021, pp. 377–387, doi: 10.1109/SANER50967.2021.00042.

[44] A. A. C. De Alwis, A. Barros, C. Fidge, and A. Polyvyanyy, "Remodularization analysis for microservice discovery using syntactic and semantic clustering," in *Advanced Information Systems Engineering*. New York, NY, USA: Springer, 2020, pp. 3–19, doi: 10.1007/978-3-030-49435-3_1.

[45] S. Rochimah and B. Nuralamsyah, "Decomposing monolithic to microservices: Keyword extraction and BFS combination method to cluster Monolithic's classes," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 7, no. 2, pp. 263–270, Mar. 2023, doi: 10.29207/resti.v7i2.4866.

[46] K. Sellami, M. A. Saied, and A. Ouni, "A hierarchical DBSCAN method for extracting microservices from monolithic applications," in *Proc. Int. Conf. Eval. Assessment Softw. Eng.*, New York, NY, USA, Jun. 2022, pp. 201–210, doi: 10.1145/3530019.3530040.

[47] M. Dehghani, S. Kolahdouz-Rahimi, M. Tisi, and D. Tamzalit, "Facilitating the migration to the microservice architecture via model-driven reverse engineering and reinforcement learning," *Softw. Syst. Model.*, vol. 21, no. 3, pp. 1115–1133, Jun. 2022, doi: 10.1007/s10270-022-00977-3.

[48] X. Sun, S. Boranbaev, S. Han, H. Wang, and D. Yu, "Expert system for automatic microservices identification using API similarity graph," *Exp. Syst.*, Oct. 2022, Art. no. e13158, doi: 10.1111/exsy.13158.

[49] M. Cojocaru, A. Uta, and A.-M. Oprescu, "MicroValid: A validation framework for automatically decomposed microservices," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2019, pp. 78–86, doi: 10.1109/CLOUDCOM.2019.00023.

[50] L. Baresi, M. Garriga, and A. De Renzis, "Microservices identification through interface analysis," in *Proc. Service-Oriented Cloud Comput.*, 2017, pp. 19–33, doi: 10.1007/978-3-319-67262-5_2.

[51] A. Selmadji, A.-D. Seriai, H. L. Bouziane, R. O. Mahamane, P. Zaragoza, and C. Dony, "From monolithic architecture style to microservice one based on a semi-automatic approach," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Mar. 2020, pp. 157–168, doi: 10.1109/ICSA47634.2020.00023.

[52] L. Nunes, N. Santos, and A. R. Silva, "From a monolith to a microservices architecture: An approach based on transactional contexts," in *Proc. Softw. Archit.*, Paris, France, Sep. 2019, pp. 37–52, doi: 10.1007/978-3-030-29983-5_3.

[53] K. Justas and M. Dalius, "Analysis of monolithic monolithic software decomposition into microservices," in *Proc. Doctoral Consortium/ForumDBIS*, 2020, pp. 25–32.

[54] D. Taibi and V. Lenarduzzi, "On the definition of microservice bad smells," *IEEE Softw.*, vol. 35, no. 3, pp. 56–62, May 2018, doi: 10.1109/MS.2018.2141031.

[55] L. J. Kirby, E. Boerstra, Z. J. C. Anderson, and J. Rubin, "Weighing the evidence: On relationship types in microservice extraction," in *Proc. IEEE/ACM 29th Int. Conf. Program Comprehension (ICPC)*, May 2021, pp. 358–368, doi: 10.1109/ICPC52881.2021.00041.

[56] L. Carvalho, A. Garcia, T. E. Colanzi, W. K. G. Assunção, J. A. Pereira, B. Fonseca, M. Ribeiro, M. J. de Lima, and C. Lucena, "On the performance and adoption of search-based microservice identification with toMicroservices," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2020, pp. 569–580, doi: 10.1109/ICSME46990.2020.00060.

[57] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Comput.*, vol. 4, no. 5, pp. 22–32, Sep. 2017, doi: 10.1109/MCC.2017.4250931.

[58] M. Hitz and B. Montazeri, "Measuring coupling and cohesion in objectoriented systems," in *Proc. Int. Symp. Appl. Corporate Comput.*, Monterrey, Mexico, 1995.

[59] P. Di Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: An industrial survey," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2018, pp. 29–2909, doi: 10.1109/ICSA.2018.00012.

[60] Y. Wang, H. Kadiyala, and J. Rubin, "Promises and challenges of microservices: An exploratory study," *Empirical Softw. Eng.*, vol. 26, no. 4, p. 63, Jul. 2021, doi: 10.1007/s10664-020-09910-y.

[61] A. K. Kalia, J. Xiao, R. Krishna, S. Sinha, M. Vukovic, and D. Banerjee, "Mono2Micro: A practical and effective tool for decomposing monolithic Java applications to microservices," in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, New York, NY, USA, Aug. 2021, pp. 1214–1224, doi: 10.1145/3468264.3473915.

[62] M. Brito, J. Cunha, and J. Saraiva, "Identification of microservices from monolithic applications through topic modelling," in *Proc. 36th Annu. ACM Symp. Appl. Comput.*, Mar. 2021, pp. 1409–1418, doi: 10.1145/3412841.3442016.

[63] S. Agarwal, R. Sinha, G. Sridhara, P. Das, U. Desai, S. Tamilselvam, A. Singhee, and H. Nakamuro, "Monolith to microservice candidates using business functionality inference," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Sep. 2021, pp. 758–763, doi: 10.1109/ICWS53863.2021.00104.

[64] A. F. A. A. Freire, A. F. Sampaio, L. H. L. Carvalho, O. Medeiros, and N. C. Mendonça, "Migrating production monolithic systems to microservices using aspect oriented programming," *Software, Pract. Exper.*, vol. 51, no. 6, pp. 1280–1307, Jun. 2021.

[65] S. Li, H. Zhang, Z. Jia, Z. Li, C. Zhang, J. Li, Q. Gao, J. Ge, and Z. Shan, "A dataflow-driven approach to identifying microservices from monolithic applications," *J. Syst. Softw.*, vol. 157, Nov. 2019, Art. no. 110380.

[66] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, 2024.
[67] J. Soldani, G. Muntoni, D. Neri, and A. Brogi, "The μTOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures," *Software, Pract. Exper.*, vol. 51, no. 7, pp. 1591–1621, Jul. 2021, doi: 10.1002/spe.2974.
[68] C. Bandara and I. Perera, "Transforming monolithic systems to microservices—An analysis toolkit for legacy code evaluation," in *Proc. 20th Int. Conf. Adv. ICT Emerg. Regions (ICTer)*, Nov. 2020, pp. 95–100, doi: 10.1109/ICTer51097.2020.9325443.
[69] T. Matias, F. F. Correia, J. Fritzsch, J. Bogner, H. S. Ferreira, and A. Restivo, "Determining microservice boundaries: A case study using static and dynamic software analysis," in *Proc. Softw. Archit.*, 2020, pp. 315–332, doi: 10.1007/978-3-030-58923-3_21.
[70] B. Liu, J. Xiong, Q. Ren, S. Tyszberowicz, and Z. Yang, "Log2MS: A framework for automated refactoring monolith into microservices using execution logs," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2022, pp. 391–396, doi: 10.1109/ICWS55610.2022.00065.
[71] A. Bucchiarone, K. Soysal, and C. Guidi, "A model-driven approach towards automatic migration to microservices," in *Proc. Softw. Eng. Aspects Continuous Develop. New Paradigms Softw. Prod. Deployment*, 2020, pp. 15–36, doi: 10.1007/978-3-030-39306-9_2.
[72] C.-Y. Li, S.-P. Ma, and T.-W. Lu, "Microservice migration using strangler fig pattern: A case study on the green button system," in *Proc. Int. Comput. Symp. (ICS)*, Dec. 2020, pp. 519–524, doi: 10.1109/ICS51289.2020.00107.
[73] A. Krause, C. Zirkelbach, W. Hasselbring, S. Lenga, and D. Kröger, "Microservice decomposition via static and dynamic analysis of the monolith," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2020, pp. 9–16, doi: 10.1109/ICSA-C50368.2020.00011.
[74] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Proc. 4th IEEE Int. Conf. Eng. Complex Comput. Syst.*, 1998, pp. 68–78, doi: 10.1109/ICECCS.1998.706657.
[75] R. Malhotra, *Rapid Java Persistence and Microservices: Persistence Made Easy Using Java EE8, JPA and Spring*. New York, NY, USA: Apress, 2019.
[76] Netflix. *Netflix Oss*. Accessed: Feb. 27, 2023. [Online]. Available: https://netflix.github.io/
[77] A. van Hoorn, J. Waller, and W. Hasselbring, "Kieker: A framework for application performance monitoring and dynamic software analysis," in *Proc. 3rd ACM/SPEC Int. Conf. Perform. Eng.*, Apr. 2012, pp. 247–248.
[78] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proc. IEEE 9th Int. Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2016, pp. 44–51, doi: 10.1109/SOCA.2016.15.
[79] I. Pigazzini, F. Arcelli Fontana, and A. Maggioni, "Tool support for the migration to microservice architecture: An industrial case study," in *Proc. Softw. Archit.*, 2019, pp. 247–263, doi: 10.1007/978-3-030-29983-5_17.
[80] D. Taibi and K. Systä, "A decomposition and metric-based evaluation framework for microservices," in *Proc. Int. Conf. Cloud Comput. Services Sci.*, Heraklion, Crete, Greece, Cham, Switzerland: Springer, 2019, pp. 133–149.
[81] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *Proc. 6th Int. Conf. Cloud Comput. Services Sci.*, 2016, pp. 137–146, doi: 10.5220/0005785501370146.
[82] M. Söylemez, B. Tekinerdogan, and A. K. Tarhan, "Challenges and solution directions of microservice architectures: A systematic literature review," *Appl. Sci.*, vol. 12, no. 11, p. 5507, May 2022, doi: 10.3390/app12115507.
[83] N. Gonçalves, D. Faustino, A. R. Silva, and M. Portela, "Monolith modularization towards microservices: Refactoring and performance trade-offs," in *Proc. IEEE 18th Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2021, pp. 1–8, doi: 10.1109/ICSA-C52384.2021.00015.
[84] R. Chen, S. Li, and Z. Li, "From monolith to microservices: A dataflow-driven approach," in *Proc. 24th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2017, pp. 466–475, doi: 10.1109/APSEC.2017.53.
[85] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Trans. Softw. Eng.*, vol. 47, no. 2, pp. 243–260, Feb. 2021, doi: 10.1109/TSE.2018.2887384. https://doi.org/10.1109/TSE.2018.2887384
[86] M. Bozkurt, M. Harman, and Y. Hassoun, "Testing and verification in service-oriented architecture: A survey," *Softw. Test., Verification Rel.*, vol. 23, no. 4, pp. 261–313, Jun. 2013, doi: 10.1002/stvr.1470.
[87] I. K. Aksakalli, T. Celik, A. B. Can, and B. Tekinerdogan, "Systematic approach for generation of feasible deployment alternatives for microservices," *IEEE Access*, vol. 9, pp. 29505–29529, 2021, doi: 10.1109/ACCESS.2021.3057582. https://doi.org/10.1109/ACCESS.2021.3057582
[88] A. A. C. De Alwis, A. Barros, A. Polyvyanyy, and C. Fidge, "Function-splitting heuristics for discovery of microservices in enterprise systems," in *Service-Oriented Computing*. Cham, Switzerland: Springer, 2018, pp. 37–53, doi: 10.1007/978-3-030-03596-9_3.
[89] F.-D. Eyitemi and S. Reiff-Marganiec, "System decomposition to optimize functionality distribution in microservices with rule based approach," in *Proc. IEEE Int. Conf. Service Oriented Syst. Eng. (SOSE)*, Aug. 2020, pp. 65–71, doi: 10.1109/SOSE49046.2020.00015.
[90] M. Kamimura, K. Yano, T. Hatano, and A. Matsuo, "Extracting candidates of microservices from monolithic application code," in *Proc. 25th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2018, pp. 571–580, doi: 10.1109/APSEC.2018.00072.
[91] F. H. Vera-Rivera, E. Puerto, H. Astudillo, and C. M. Gaona, "Microservices backlog—A genetic programming technique for identification and evaluation of microservices from user stories," *IEEE Access*, vol. 9, pp. 117178–117203, 2021, doi: 10.1109/ACCESS.2021.3106342.
[92] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *J. Syst. Softw.*, vol. 150, pp. 77–97, Apr. 2019, doi: 10.1016/j.jss.2019.01.001.
[93] A. Razzaq, "A systematic review on software architectures for IoT systems and future direction to the adoption of microservices architecture," *Social Netw. Comput. Sci.*, vol. 1, no. 6, p. 350, Oct. 2020, doi: 10.1007/s42979-020-00359-w.
[94] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *J. Syst. Softw.*, vol. 182, Dec. 2021, Art. no. 111061, doi: 10.1016/j.jss.2021.111061.
[95] M. F. Gholami, F. Daneshgar, G. Low, and G. Beydoun, "Cloud migration process—A survey, evaluation framework, and open challenges," *J. Syst. Softw.*, vol. 120, pp. 31–69, Oct. 2016, doi: 10.1016/j.jss.2016.06.068.
[96] J. Ghofrani and D. Lübke, "Challenges of microservices architecture: A survey on the state of the practice," in *Proc. ZEUS*, 2018, pp. 1–8.
[97] A. Christoforou, L. Odysseos, and A. Andreou, "Migration of software components to microservices: Matching and synthesis," in *Proc. 14th Int. Conf. Eval. Novel Approaches to Softw. Eng.*, 2019, pp. 134–146, doi: 10.5220/0007732101340146.
[98] Z. Ren, W. Wang, G. Wu, C. Gao, W. Chen, J. Wei, and T. Huang, "Migrating web applications from monolithic structure to microservices architecture," in *Proc. 10th Asia–Pacific Symp. Internetware*, 2018, pp. 1–12, doi: 10.1145/3275219.3275230.
[99] O. Al-Debagy and P. Martinek, "Extracting Microservices' candidates from monolithic applications: Interface analysis and evaluation metrics approach," in *Proc. IEEE 15th Int. Conf. Syst. Syst. Eng. (SoSE)*, Jun. 2020, pp. 289–294, doi: 10.1109/SoSE50414.2020.9130466.
[100] M. M. Lehman, "Laws of software evolution revisited," in *Proc. Softw. Process Technol.*, 1996, pp. 108–124, doi: /10.1007/BFb0017737.
[101] H. Jain, H. Zhao, and N. R. Chinta, "A spanning tree based approach to identifying web services," *Int. J. Web Services Res.*, vol. 1, no. 1, pp. 1–20, Jan. 2004, doi: 10.4018/jwsr.2004010101.
[102] M. P. Robillard, E. Bodden, D. Kawrykow, M. Mezini, and T. Ratchford, "Automated API property inference techniques," *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 613–637, May 2013, doi: 10.1109/TSE.2012.63. https://doi.org/10.1109/TSE.2012.63
[103] K. Manikas, "Revisiting software ecosystems research: A longitudinal literature study," *J. Syst. Softw.*, vol. 117, pp. 84–103, Jul. 2016, doi: 10.1016/j.jss.2016.02.003.
[104] M. Abdullah, W. Iqbal, and A. Erradi, "Unsupervised learning approach for web application auto-decomposition into microservices," *J. Syst. Softw.*, vol. 151, pp. 243–257, May 2019, doi: 10.1016/j.jss.2019.02.031.
[105] S. Scalabrino, G. Bavota, M. Linares-Vásquez, M. Lanza, and R. Oliveto, "Data-driven solutions to detect API compatibility issues in android: An empirical study," in *Proc. IEEE/ACM 16th Int. Conf. Mining Softw. Repositories (MSR)*, May 2019, pp. 288–298, doi: 10.1109/MSR.2019.00055.

[106] H. Knoche and W. Hasselbring, "Using microservices for legacy software modernization," *IEEE Softw.*, vol. 35, no. 3, pp. 44–49, May 2018, doi: 10.1109/MS.2018.2141035.
[107] A. A. C. De Alwis, A. Barros, C. Fidge, and A. Polyvyanyy, "Microservice remodularisation of monolithic enterprise systems for embedding in industrial IoT networks," in *Proc. Adv. Inf. Syst. Eng.*, 2021, pp. 432–448, doi: 10.1007/978-3-030-79382-1_26.
[108] D. Bajaj, U. Bharti, A. Goel, and S. C. Gupta, "Partial migration for re-architecting a cloud native monolithic application into microservices and FaaS," in *Proc. Inf., Commun. Comput. Technol.*, 2020, pp. 111–124, doi: 10.1007/978-981-15-9671-1_9.
[109] M. Liu, X. Peng, A. Marcus, Z. Xing, W. Xie, S. Xing, and Y. Liu, "Generating query-specific class API summaries," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 120–130, doi: 10.1145/3338906.3338971.
[110] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, "Service cutter: A systematic approach to service decomposition," in *Proc. Service-Oriented Cloud Comput.*, 2016, pp. 185–200, doi: 10.1007/978-3-319-44482-6_12.
[111] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Inf. Softw. Technol.*, vol. 85, pp. 43–59, May 2017, doi: 10.1016/j.infsof.2017.01.006.

IDRIS OUMOUSSA received the bachelor's degree in software engineering from Ibn Zohr University, in 2018. He is currently a Software Engineer with a passion for modernizing IT landscapes, delves into the complexities of microservices architectures with the National Institute of Statistics and Applied Economics (INSEA), Morocco, as a Ph.D. Researcher. His research interests include microservice architectures, cloud computing, component-based software, artificial intelligence, and agile software development.

RAJAA SAIDI received the dual Ph.D. degree in information systems and software engineering from Mohammed V University (UM5), Rabat, Morocco, and the Grenoble Institute of Technology (INPG), France. She is currently a Full Professor of computer science with the National Institute of Statistics and Applied Economics (INSEA), Morocco, where she is a member with the Information Systems, Intelligent Systems and Mathematical Modelling Laboratory (SI2M). Her research interests include information systems engineering, business process management, ubiquitous computing, context-aware information systems, service-oriented architectures, and component-based engineering.