A Qualitative Literature Review on Microservices Identification Approaches

Christoph Schröer(✉), Felix Kruse, and Jorge Marx Gómez

Department Very Large Business Applications, University of Oldenburg, 26129 Oldenburg, Germany
{christoph.schroeer,felix.kruse,jorge.marx.gomez}@uol.de

Abstract. Microservices has become a widely used and discussed architectural style for designing modern applications due to advantages like granular scalability and maintainability. However, it is still a complex task decomposing an application into microservices. Software architects often design architectures manually. In this paper we give a state-of-the-art overview of current approaches to identifying microservices. Therefore we use a literature review and classify the content based on the software development process.

The main results are that mostly monolithic artifacts are used for starting with microservice decomposition. Data-intensive applications are less focused. Rule-based and clustering algorithms are suitable ways to find microservice candidates. Both researchers and software architects profit from this overview. Practically it supports choosing suitable approaches considering aspects like cohesion, coupling, workload, deployment and further quality criteria for each phase during the software development process.

Keywords: Microservices · Literature review · Software architecture · Identification · Decomposition

1 Introduction

Microservices become a popular architectural style for designing modern applications and deploying these on cloud platforms [7]. They are defined as small services which are independently and continuously deployable and communicating through lightweight mechanisms. Specific characteristics are their high cohesion and loose coupling [18]. This kind of architecture addresses beneficial quality attributes like scalability and maintainability, but also has drawbacks and can end-up in higher complexity [7,10]. In contrast, monolithic applications are the traditional architectural style. Monoliths are usually limited to only one technology and require higher effort on maintenance, but are less complex [18].

Architecting microservices has several perspectives. Besides infrastructural and organizational aspects, designing the services itself has become increasing attention in research. For example, research focuses on finding a suitable size,

S. Dustdar (Ed.): SummerSOC 2020, CCIS 1310, pp. 151–168, 2020.
https://doi.org/10.1007/978-3-030-64846-6_9

granularity and number of microservices. This is a complex challenge because several criteria like cohesion, coupling and quality attributes influence design decisions. Furthermore, there is a runtime uncertainty. Too much microservices can result in communication overhead, whereas too less microservices can retain the drawbacks of monoliths [3,14]. Figure 1 shows a visual example of the influence of the number of microservices on the costs of deployment and quality assurance.

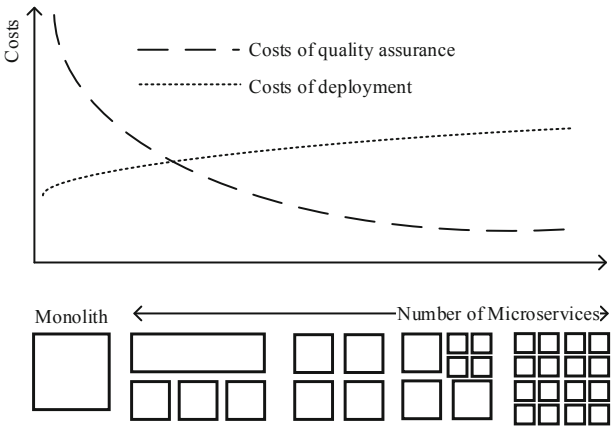

Fig. 1. Influence of the number of microservices on cost criteria [12].

Domain-driven design is an often mentioned approach for designing microservices [24]. However, more approaches exist. The approaches differ in focused criteria, different algorithms and evaluation methods. Due to these many and manifolds approaches, it is challenging to choose an appropriate method. Therefore, this study aims to conduct a literature review of existing approaches for identifying microservices. The approaches are classified into the phases of the software development process: requirements analysis, design phase, implementation phase and test phase. Microservices can be developed within the software development process [7] so that our new classification approach helps to find methods for each phase during the development and evaluation of microservices. This study has practical and theoretical implications. Practically, software architects can select appropriate methods from literature. Theoretically, researchers profit from a recent overview of existing approaches.

The study is structured as follows. Section 2 gives an overview of related work that can be compared to our literature review. Section 3 explains the research methodology and defines the research questions. Section 4 presents the results. Section 5 discusses the results and Sect. 6 sums up and gives an outlook of further research.

2 Related Work: Literature Reviews on Approaches for Identifying Microservices

Architecting microservices means to decompose a software application into a set of small services. The topic of software decomposition is not new to microservices and are also related to identifying modules and components in (monolithic) application in general. Different approaches like weighted-clustering approaches can support the software decomposition process [1]. However, further criteria need to be integrated into microservices architecture [21]. There are existing literature reviews for microservices architectures. These reviews focus on infrastructure, architectures and target problems and do not research the design phase of microservices in detail [8,26].

Cojocaru [6] conducted a literature review and evaluated specific quality criteria on static and dynamic source code analysis. However, they focused on decomposition of monoliths alone, whereas we extend our search query and are not limited to monoliths.

The study of Kazanavicius [16] summarized an authors selection of different approaches for migrating legacy monolithic applications to microservices architecture. The authors assessed quality, benefits and drawbacks of the approaches. However, they only compared and summarized a small number of studies.

Fritzsch [11] selected studies by a search query focused on refactoring monolithic applications. The classification framework was developed from the selected studies inductively and focused on different strategies. However, the classification framework of this paper considers the software development process and is not limited on refactoring.

3 Methodology: Qualitative Content Analysis

For this study, we choose the method of qualitative content analysis based on research literature. Qualitative content analysis is an appropriate research method in design science research [27]. The qualitative content analysis considering scientific publications and studies is a literature review. Mayring [20] defines the qualitative content analysis as a "mixed methods approach: assignment of categories to text as qualitative step, working through many text passages and analysis of frequencies of categories as quantitative step" [20]. Following its exploratory research design, the steps are (1) defining the research questions, (2) searching the data, (3) formulating categories inductively and (4) proofing them after approx. 15% of the data. Following steps are (5) the processing of the full study and (6) discussing the results [5,20].

In general, we would like to answer the main research question:

- Which approaches do exist to support the identification of microservices along the software development process considering several criteria?

Therefore, we concretize the research questions along the software development process:

- RQ1: Which starting points can be used for microservices identification approaches during requirements analysis?
- RQ2: Which microservices identification approaches do exist at design phase?
- RQ3: How can identified microservices be evaluated?

To operationalize the research questions, we formulate a search strategy based on the research questions. The different terms of writing microservice are based on DiFrancesco [8]. Furthermore, we find synonyms for "identification" in the basic literature book written by Richardson [24] and adjust or extend terms after 15% of the data. Finally, we applied the following search strategy in relevant scientific data sources ScienceDirect, IEEE and ACM Digital Library in the beginning of 2020:

ALL = ((microservice OR microservices) AND (identif* OR extract* OR migrati* OR decompos* OR refactor*)) AND PY >= (2014)

We have chosen to limit the results regarding the publication year to 2014 since the concept of microservices has established since 2014 [18]. We define the inclusion criteria according to which the studies must describe concrete approaches for identification and be written in English. We exclude papers that are out of scope, are grey literature and have no abstracts. The method of snowballing (see [23]) results in four additional paper. The number of resulted studies are shown in Fig. 2. Appendix A lists the selected paper. Totally, we have read 81 abstracts of all found paper and 31 full texts of relevant paper.

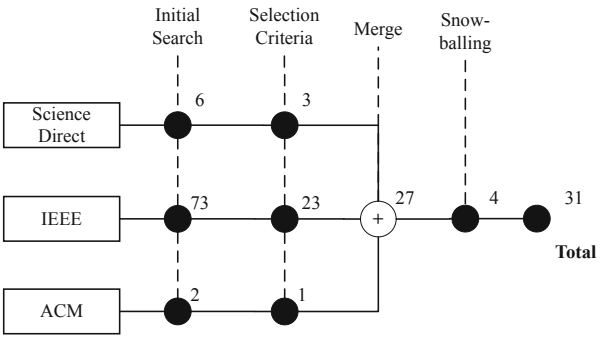

Fig. 2. Search results.

For the classification framework, we developed three main categories inductively that follow the software development process: requirement analysis, design phase and implementation/evaluation phase. We exclude the operation phase because the identification of microservices is mainly a design than an opreational topic. For answering the research question, this phase is out of scope. During reading the full texts, we continuously develop sub-categories deductively from the papers [20].

4 State-of-the-Art Identification Approaches for Microservices

The number of publications grouped by publication year are shown in Fig. 3. There is a higher number since 2017, but a trend for increasing number of studies is not observable.

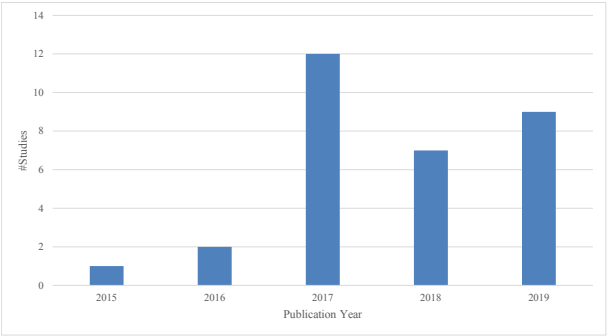

Fig. 3. Number of the selected papers grouped by publication year.

4.1 RQ1: Requirements Analysis of Microservice Identification Approaches

Accordingly to [7], we also identified that the main research focus is migration from monoliths to microservices architecture. Therefore, 21 of the 31 papers start with artifacts from monoliths. For static artifacts, 17 studies deal with source code, API documentation, domain models or data from version control systems. For dynamic analysis, four studies start with execution logs of the monolith. Table 1 lists further results.

Table 2 shows that there is a focus on *general, transactional-oriented applications* like web and mobile applications. For *data-intensive applications*, three papers propose a method for decomposition in microservices.

4.2 RQ2: Microservice Identification Approaches for Design Phase

The design phase is crucial in software development process in general. In this phase, the architecture of a system is structured in several components and their dependencies [25].

Di Francesco [7] have found out that the main research focus on architecting microservices is design phase. The following findings go into detail regarding the decomposition approaches.

Extending the idea of Fritzsch [11], we also consider atomar units. We define an atomar unit as the smallest unit on that a decomposition approach begins.

Table 1. Starting points for identification.

Starting points for identification	Details	#	Studies
Monolith (static)	e.g. source code, migration, API specification, domain model	17	P27, P26, P25, P24, P23, P22, P19, P15, P14, P12, P9, P8, P7, P5, P4, P3, P1
Greenfield	e.g. list of requirements	9	P30, P29, P28, P21, P17, P13, P6, P4, P2
Business processes	e.g. BPMN	5	P24, P18, P16, P11, P6
Monolith (dynamic)	e.g. execution logs	4	P31, P25, P20, P10
Monolith first	like greenfield, but designing a monolith first	2	P21, P20

Table 2. Type of application.

Type of application	Details	#	Studies
General applications	e.g. web or mobile applications, transaction-oriented applications	26	P31, P30, P27, P25, P24, P23, P22, P21, P20, P19, P18, P16, P15, P14, P12, P11, P10, P9, P8, P7, P6, P5, P4, P3, P2, P1
Data-intensive applications	e. g. IoT, machine learning, stream	3	P29, P28, P17
Telecommunication/Network		2	P26, P13

We identified seven types of atomar units, as Table 3 shows. Plain *functions* are the most frequent atomar units. For example, function calls can be logged. Such logs of function executions can be used to group microservices (see P20).

Modeling the Application. During the design phase, the application has to be modeled with suitable approaches and modeling techniques. These models are the basis for processes or algorithms for finding microservice candidates. We differ between modeling approaches, the algorithms and the criteria that influence the service decomposition. The Tables 4, 5, 6 and 7 contain these topics.

Table 3. Atomar units.

Atomar unit	Details	#	Studies
Function	e.g. public methods of service classes, functions or properties	8	P29, P28, P27, P26, P21, P20, P10, P8
Business capability	e. g. business activity in BPMN, business vocabulary based on nouns	7	P30, P18, P16, P14, P11, P9, P6
Interface	e. g. facades, entry points, URI resources	6	P31, P22, P19, P15, P4, P1
Entity	e.g. also database tables	4	P12, P9, P3, P1
Functionality	set of functions; functional units, atoms, modules	3	P25, P13, P7
With one mention		2	requirement (P2), software tier (P17)
n.a./not clear		3	P24, P23, P5

Table 4 lists the identified modeling approaches. We defined a category called *decomposition through atomar units* (six studies). That means that the prior identified atomar units are used for decomposition. For example, Tserpes (2019) defines functions as atomar units and develops each of these functions as an own microservice (see P29). Further, the *domain-driven design approach* is also part of research in this context. For example, Knoche et al. (2019) describe a process of defining bounded context focused on microservices (see P22).

Graph-based approaches are used to model atomar units and their dependencies as directed graphs. For example, Kamimura et al. (2018) model functions and function-calls as a graph (see P19). Similarly, *data-flow driven approaches* represent applications through data flow diagrams to model data dependencies. For example, Li et al. (2019) model data exchange of a ticket booking system with data-flow diagrams (see P30).

The *black-box approach* uses interfaces (not specific classes or entities) as the entry point of applications. For example, Baresi et al. (2017) use OpenAPI specifications and calculate semantic similarities to group interfaces into microservices (see P15). *Performance models* are used to consider workload criteria. For example, Klock et al. (2017) model a M/M/1 queuing network for a microservice performance model (see P10).

The *manual nominal/ordinal estimation model* evaluates specific requirements of an application regarding their importance manually. For example, Shimoda et al. (2018) evaluate the possibility to change of functions on a scale between 0 and 5 (see P21).

Table 4. Approaches/models.

Approach/models	#	Studies
Modeling through atomar units	6	P29, P27, P24, P13, P7, P2
Scenario analysis	5	P24, P18, P14, P5, P1
Graph-based approach	5	P19, P10, P8, P3, P1
Domain-driven design	4	P24, P22, P14, P12
Dataflow-driven approach	4	P30, P16, P11, P9
Black-box-approach	3	P31, P15, P4
Tier-based approach	3	P17, P7, P1
Performance model	3	P28, P26, P10
Execution-trace modeling	3	P25, P20, P1
Manual nominal/ordinal estimation model	2	P21, P2
With one mention	2	probabilistically (P6), interview (P23)

Identifying Microservice Candidates. After modeling the application, specific algorithms and methods are used to identify microservices candidates (see Table 5). *Rule-based algorithms* are the most popular method, followed by *clustering algorithms*. One example for a rule-based approach is presented in P22 where deterministic rules are used from defining services facades covering functionalities to their mapping to microservices. Based on a graph model, the authors of P8 use a minimum spanning tree based on weights of the edges and build then clusters with deleting remaining edges by depth-first-search. Three studies define the identification of microservices as a multi-criteria optimization problem and use *genetic algorithms* for solving.

One finding is that eight studies give hints on microservice identification *informally*, but not in form of a particular method or algorithm. We have not classified them as irrelevant because these studies also give valuable support for microservice identification. For example, Gouigoux et al. (2017) explain in their experience report relationships of costs of service deployments and service granularity (see P14).

Table 5. Algorithms and methods.

Algorithm/Method	#	Studies
Rule-based algorithm	13	P30, P29, P28, P27, P22, P21, P19, P18, P17, P9, P4, P2, P1
Unsupervised learning/clustering	9	P31, P25, P20, P19, P16, P10, P8, P6, P3
Genetic algorithm	3	P26, P25, P10
Natural language processing	1	P15
n.a./manually/informal	8	P24, P23, P14, P13, P12, P11, P7, P5

Criteria for Identification of Approaches. The proposed approaches and algorithms from all 31 studies differ in the criteria used for decomposition. Accordingly to the qualitative study of Carvalho et al. (2019), we also identified criteria like cohesion, coupling, workload and quality attributes [4]. Further, we have quantified the occurrence.

Table 6 shows *cohesion* and *coupling* criteria. Seventeen of the research papers includes cohesion and coupling criteria in their approaches. Following the definition by Richardson [24] it matters "[...] that each service has a focused, cohesive set of responsibilities.". Cohesive criteria can be characterized by the atomar units itself (see Table 3), by specific metrics, by business capabilities or by interface similarity.

Table 6. Cohesion and coupling criteria.

Cohesion	Details	#	Studies
Atomar unit oriented	e.g. ServiceClass public methods of service classes, functions or properties in general	6	P29, P28, P27, P24, P13, P9
Cohesion metric	Relational cohesion, Cohesion at domain level/message level, resource count	4	P30, P25, P20, P4
Business capabilities	e. g. facades, entry points, URI resources	4	P24, P18, P5, P1
Semantic or reference similarity		3	P24, P15, P3
Qualitative		6	P15, P13, P12, P11, P9, P3
Coupling	Details	#	Studies
Dependencies between atomar units	Foreign keys, intra/inter-connectivity	10	P25, P19, P16, P12, P10, P8, P7, P3, P2, P1
Coupling metric	Afferent/efferent coupling, instability, interactions, structural/ conceptual modularity quality	4	P30, P25, P20, P4
TF/IDF		1	P8
Qualitative		5	P15, P13, P12, P11, P9, P3

Coupling criteria consider the dependencies of the atomar units. For example, Kamimura et al. (2018) use source code dependencies of classes to extract microservices (see P19). Further, concrete metrics can measure coupling. The *qualitative* row means that cohesion and coupling are taken into consideration, but the authors have not described that precisely that it can be classified here.

Table 7. Further criteria adopted in the selected studies.

Criteria	Sub-criteria	#	#Studies
Deployment and infrastructure		8	P31, P28, P26, P21, P17, P14, P13, P7
Workload		5	P31, P26, P10, P7, P3
Compatibility		2	P7, P3
Quality attributes	Change frequency	6	P25, P21, P8, P7, P6, P3
	Scalability	4	P21, P6, P5, P2
	Data coordination effort	3	P21, P19, P3
	Organisation	2	P8, P5
	Suitable technologies	2	P7, P6
	With one mention	4	Cost (P14), Variability (P23), Security (P2), Reliability (P5)

Table 7 lists non-functional criteria for decomposition into microservices. *Deployment criteria* mean that the automation of deployment (like in P14) or deployment on different computing instances (like in P28) influence the decomposition into microservices. *Workload criteria* consider expected response times, count of requests or document sizes. *Compatibility* (like P3) defines a microservice with compatible criteria, like compatible technologies. *Quality attributes* are divided into further sub-criteria like common changes, scalability, data coordination effort, team organization and suitable technologies. The table aggregates criteria that have been mentioned one time.

Finally, we will give one example, how to connect the categories of the design phase. Giving the paper P25. the tables show that the approach includes coupling criteria. Further, the approach models the application on execution-trace modeling based on the atomar unit of functionality. To find optimal microservice candidates, the authors propose a genetic algorithm.

4.3 RQ3: Testing and Evaluation of Microservice Identification Approaches

It is crucial that authors evaluate their proposed approaches. Primarily, we differ between *runtime* evaluation and *case study* evaluation, whereas the latter is a theoretical evaluation without implemented microservices (see Table 8). The

latter occurred 18 times, the former seven times. We classified four papers as an *experience report* since the authors evaluated the decomposition in a real-world application. Only two papers do not describe an evaluation of their approaches in the published study.

Runtime evaluation concentrates on performance metrics like:

- throughput (P31, P29, P17, P13, P10),
- response time (P31, P26, P17, P10),
- CPU utilization (P31, P26, P13),
- data communication over network (P28, P17),
- runtime costs (P31, P29),
- memory usage (P13),
- allocated virtual machines (P31) and
- total execution time over all microservices (P28).

Table 8. Evaluation methods.

Evaluation methods	#	Studies
Case study	18	P30, P27, P25, P21, P20, P19, P18, P16, P15, P12, P11, P9, P8, P6, P4, P3, P2, P1
Experiments	7	P31, P29, P28, P26, P17, P13, P10
Experience report	4	P24, P22, P14, P5
n.a	2	P23, P7

The case studies identify microservices candidates and evaluate resulted architectures with:

- experts (P19, P16, P15, P3),
- with required time for decomposition process (P27, P8, P3) or
- compare the number of resulted number of microservice candidates (P27, P19, P2, P1).

A further finding is that the application code used for evaluation are mostly *non-public or own developed* (Table 9). Eleven papers use *public available* source code basis. For example, the authors of P30 use the Cargo Tracking System (https://github.com/citerus/dddsample-core) that initially was developed for illustrating domain-driven design by Evans [9].

Table 9. Applications for evaluation.

Evaluation application	#	Studies
own developed/not public	22	P29, P28, P27, P26, P24, P23, P22, P19, P18, P17, P16, P14, P12, P10, P9, P8, P7, P6, P5, P4, P2, P1
Public available	11	P31, P30, P25, P21, P20, P19, P15, P13, P11, P8, P3

Third, we also investigate how the resulted microservice architectures are compared (Table 10). Twelve papers compare their microservice architectures with *alternative microservice architectures* decomposed by another approach. Four papers compare a microservice architecture with the monolithic application as *baseline*. The majority of the paper do not compare any decomposed architectures.

5 Interpretation of the Results and the Derived Future Research Topics

The purpose of the literature review is to give the current state-of-the-art methods of microservice identification approaches. Decomposing an application into smaller parts is a challenging task in software development in general. However, microservice development adds more complexity due to new aspects like runtime processes and team organization. Finally, we identified the following four main findings.

Table 10. Comparisons.

Comparison	#	Studies
Between microservice architectures	12	P31, P30, P26, P25, P20, P19, P17, P16, P15, P10, P9, P8
Compared with monolith	4	P29, P26, P24, P15
No comparison	16	P28, P27, P23, P22, P21, P18, P14, P13, P12, P11, P7, P6, P5, P4, P2, P1

1. Focus on Monoliths: Our results show that for the identification of microservices mostly monolithic artifacts are the standard starting point. One reason could be that monoliths have already existing artifacts like source code or runtime logs. Starting from greenfield development directly could also have limitations like higher complexity at the beginning. However, approaches for greenfield development regarding microservices identification have still less attention in research, as Table 1 shows.

As the identification of microservice is a non-trivial challenge, approaches considering also the new development of applications, can facilitate the usage of microservice architectures. Potential artifacts can be lists of requirements, business oriented documents or also interviews. However, the potential of usage such artifacts should be part of further research.

2. Formal and Informal Methods Exist: Table 5 also shows that eight papers have proposed a non-structural way of finding microservice candidates. Further research should be done regarding structured algorithms and non-informal, comparable processes to support software architects in a more formally way. The Tables 6 and 7 show the manifold criteria catalog that could be considered during the development of microservices.

If the methods for the identification are not sufficient, the perspective of service-oriented architectures (SOA) can be included, as microservices are a kind of SOA. In this field, existing approaches like Web Services Development Lifecycle (SDLC) and Service-Oriented Modeling and Architecture (SOMA) integrates a service modeling and service identification phase. These phases can support the design of microservices, as the services from SOA have to be mapped to microservices [2,15,22,28].

3. Experimental Evaluations are Underrepresented: The results of RQ3 shows further that only seven times papers conducted experiments, although the granularity of services has a significant impact regarding the performance of the whole system. We think, that experiments could strengthen the evaluation of microservice identification approaches. The evaluation with case studies and with experts seems that manual effort is further necessary.

4. Data-Intensive Applications Mostly Out of Scope: We also observed that identification approaches were developed mostly for general, transaction-oriented applications (such as web or mobile applications), as Table 2 shows. Data-intensive applications have not been considered yet, although data science, artificial intelligence and big data applications are strongly in the scientific and economic focus.

Microservices could be a suitable architecture for data-intensive application to integrate different technologies, algorithms and applications. Data-intensive applications are analytical-oriented. Mostly, they implement so-called data pipe"-lines. The requirements of such applications differ from those of transactional applications due to the characteristics of Big Data (5 Vs). With further research in this direction, software architects could be significantly supported in the development of data-intensive applications based on microservice architecture.

Existing architectural patterns for implementing data-intensive applications are batch, stream and lambda architecture pattern [19]. These patterns are a modular approach, that could be combined with the microservice architecture style. First approaches for using service-oriented approaches or microservices in data-intensive context exist for the processing of large geospatial data in the cloud [17] or for entity matching approaches for data integration [13]. However, those approaches concentrate on specific tasks, domains or technologies. Further research should also focus on how microservices can be identified.

6 Conclusion

This paper explores the recent microservice identification approaches. These approaches are necessary to find suitable microservices considering several criteria like cohesion, coupling and quality attributes. However, one limitation is that we have not evaluated the classification framework in practice yet. We can only

assume the practical implications, although we think that the results could support software architects in the choice of appropriate approaches. Furthermore, the deductive derivation of the categories and the selection of relevant papers are based on our assertions or other publications.

Based on a comprehensive qualitative literature review, an overview of the state-of-the-art approaches is presented. A total of 31 papers were selected as relevant. We have classified the microservice development into the phases of the software development process. We derived further categories deductively. The results are structured into the initially phrases research questions. Answering RQ1, mostly monolithic artifacts are used for starting with microservice decomposition. Answering RQ2, formal and informal approaches exist. Following formal approaches, modern algorithms like clustering algorithms could also an appropriate way finding microservice candidates. Answering RQ3, experimental evaluations for measuring performance metrics of microservice architecture are underrepresented.

The practical implication is that software architects could select a suitable approach regarding pre-defined criteria. The theoretical implication is the state-of-the-art overview for researchers. One result is that microservice identification approaches for data-intensive and analytical applications are underrepresented. We want to deal with this problem in the future and make recommendations for finding suitable microservices in a data-intensive context.

A Selected Studies

The selected studies are listed below and in reverse chronological order.

Key	Study
P31	Abdullah, M., Iqbal, W., Erradi, A.: Unsupervised learning approach for web application auto-decomposition into microservices. Journal of Systems and Software (2019)
P30	Li, S., Zhang, H., Jia, Z., Li, Z., Zhang, C., Li, J., Gao, Q., Ge, J., Shan, Z.: A dataflow-driven approach to identifying microservices from monolithic applications. Journal of Systems and Software (2019)
P29	Tserpes, K.: stream-MSA: A microservices' methodology for the creation of short, fast-paced, stream processing pipelines. ICT Express (2019)
P28	Alturki, B., Reiff-Marganiec, S., Perera, C., De, S.: Exploring the Effectiveness of Service Decomposition in Fog Computing Architecture for the Internet of Things. IEEE Transactions on Sustainable Computing (2019)
P27	Kaplunovich, A.: ToLambda-Automatic Path to Serverless Architectures. In: 2019 IEEE/ACM 3rd International Workshop on Refactoring (IWoR), pp. 1–8 (2019)

(continued)

Table 1. *(continued)*

Key	Study
P26	Sharma, S., Uniyal, N., Tola, B., Jiang, Y.: On Monolithic and Microservice Deployment of Network Functions. In: 2019 IEEE Conference on Network Softwarization (NetSoft), pp. 387–395 (2019)
P25	Jin, W., Liu, T., Cai, Y., Kazman, R., Mo, R., Zheng, Q.: Service Candidate Identification from Monolithic Systems based on Execution Traces. IEEE Transactions on Software Engineering (2019)
P24	Gouigoux, J.; Tamzalit, D.: "Functional-First" Recommendations for Beneficial Microservices Migration and Integration Lessons Learned from an Industrial Experience. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 182–186 (2019)
P23	Carvalho, L., Garcia, A., Assunção, W.K.G., Bonifácio, R., Tizzei, L.P., Colanzi, T.E.: Extraction of Configurable and Reusable Microservices from Legacy Systems: An Exploratory Study. In: Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A, pp. 26–31. Association for Computing Machinery, New York, NY, USA (2019)
P22	Knoche, H., Hasselbring, W.: Using Microservices for Legacy Software Modernization. IEEE Softw. (2018)
P21	Shimoda, A., Sunada, T.: Priority Order Determination Method for Extracting Services Stepwise from Monolithic System. In: 2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 805–810 (2018)
P20	Jin, W., Liu, T., Zheng, Q., Cui, D., Cai, Y.: Functionality-Oriented Microservice Extraction Based on Execution Trace Clustering. In: 2018 IEEE International Conference on Web Services (ICWS), pp. 211–218 (2018)
P19	Kamimura, M., Yano, K., Hatano, T., Matsuo, A.: Extracting Candidates of Microservices from Monolithic Application Code. In: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), pp. 571–580 (2018)
P18	Tusjunt, M., Vatanawood, W.: Refactoring Orchestrated Web Services into Microservices Using Decomposition Pattern. In: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), pp. 609–613 (2018)
P17	Sriraman, A., Wenisch, T.F.: μ Suite: A Benchmark Suite for Microservices. In: 2018 IEEE International Symposium on Workload Characterization (IISWC), pp. 1–12 (2018)
P16	Amiri, M.J.: Object-Aware Identification of Microservices. In: 2018 IEEE International Conference on Services Computing (SCC), pp. 253–256 (2018)
P15	Baresi, L., Garriga, M., Renzis, A. de: Microservices Identification Through Interface Analysis. In: Paoli, F. de, Schulte, S., Broch Johnsen, E. (eds.) Service-Oriented and Cloud Computing. European Conference, ESOCC 2017, Oslo, Norway, September 27–29, 2017 (2017)
P14	Gouigoux, J.; Tamzalit, D.: From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 62–65 (2017)

(continued)

Table 1. *(continued)*

Key	Study
P13	Boubendir, A., Bertin, E., Simoni, N.: A VNF-as-a-service design through micro-services disassembling the IMS. In: 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), pp. 203–210 (2017)
P12	Fan, C., Ma, S.: Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report. In: 2017 IEEE International Conference on AI Mobile Services (AIMS), pp. 109–112 (2017)
P11	Hausotter, A., Koschel, A., Zuch, M., Busch, J., Kreczik, A.: Process and Service Modelling of Insurancy Use Cases. In: 2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA), pp. 116–124 (2017)
P10	Klock, S., van der Werf, J.M.E.M., Guelen, J.P., Jansen, S.: Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures. In: 2017 IEEE International Conference on Software Architecture (ICSA), pp. 11–20 (2017)
P9	Chen, R., Li, S., Li, Z.: From Monolith to Microservices: A Dataflow-Driven Approach. In: 2017 24th Asia-Pacific Software Engineering Conference (APSEC), pp. 466–475 (2017)
P8	Mazlami, G., Cito, J., Leitner, P.: Extraction of Microservices from Monolithic Software Architectures. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 524–531 (2017)
P7	Sarita, Sebastian, S.: Transform Monolith into Microservices using Docker. In: 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), pp. 1–5 (2017)
P6	Sayara, A., Towhid, M.S., Hossain, M.S.: A probabilistic approach for obtaining an optimized number of services using weighted matrix and multidimensional scaling. In: 2017 20th International Conference of Computer and Information Technology (ICCIT), pp. 1–6 (2017)
P5	Hasselbring, W., Steinacker, G.: Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 243–246 (2017)
P4	Asik, T., Selcuk, Y.E.: Policy enforcement upon software based on microservice architecture. In: 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 283–287 (2017)
P3	Gysel, M., Kölbener, L., Giersche, W., Zimmermann, O.: Service Cutter: A Systematic Approach to Service Decomposition. In: Aiello, M., Johnsen, E.B., Dustdar, S., Georgievski, I. (eds.) Service-Oriented and Cloud Computing. 5th IFIP WG 2.14 European Conference, ESOCC 2016, Vienna, Austria, September 5-7, 2016, Proceedings, pp. 185–200 (2016)
P2	Ahmadvand, M., Ibrahim, A.: Requirements Reconciliation for Scalable and Secure Microservice (De)composition. In: 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), pp. 68–73 (2016)
P1	Levcovitz, A., Terra, R., Valente, M.T.: Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems. 3rd Brazilian Workshop on Software Visualization, Evolution and Maintenance (VEM), 97–104 (2015)

References

1. Andritsos, P., Tzerpos, V.: Information-theoretic software clustering. IEEE Trans. Softw. Eng. **31**(2) (2005)
2. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. **47**(3), 377–396 (2008)
3. Carrasco, A., van Bladel, B., Demeyer, S.: Migrating towards microservices: migration and architecture smells. In: Proceedings of the 2nd International Workshop on Refactoring, IWoR 2018, pp. 1–6. Association for Computing Machinery, New York (2018)
4. Carvalho, L., Garcia, A., Assunção, W.K.G., Mello, R.d., Lima, M.J.d.: Analysis of the criteria adopted in industry to extract microservices. In: 2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER IP), pp. 22–29 (2019)
5. Cato, P.: Einflüsse auf den Implementierungserfolg von Big Data Systemen: Ergebnisse einer inhalts- und kausalanalytischen Untersuchung. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen-Nürnberg (2016)
6. Cojocaru, M., Uta, A., Oprescu, A.: Attributes assessing the quality of microservices automatically decomposed from monolithic applications. In: 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), pp. 84–93 (2019)
7. Di Francesco, P., Lago, P., Malavolta, I.: Architecting with microservices: a systematic mapping study. J. Syst. Softw. **150**, 77–97 (2019)
8. Di Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: trends, focus, and potential for industrial adoption. In: IEEE International Conference on Software Architecture, pp. 21–30 (2017)
9. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, Upper Saddle River (2011)
10. Fritzsch, J., Bogner, J., Wagner, S., Zimmermann, A.: Microservices migration in industry: intentions, strategies, and challenges. In: IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 481–490 (2019)
11. Bruel, J.-M., Mazzara, M., Meyer, B. (eds.): DEVOPS 2018. LNCS, vol. 11350. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-06019-0
12. Gouigoux, J., Tamzalit, D.: From monolith to microservices: lessons learned on an industrial migration to a web oriented architecture. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 62–65 (2017)
13. Govind, Y., et al.: Entity matching meets data science. In: SIGMOD 2019: Proceedings of the 2019 International Conference on Management of Data, pp. 389–403 (2019)
14. Hassan, S., Bahsoon, R.: Microservices and their design trade-offs: a self-adaptive roadmap. In: IEEE International Conference on Services Computing, pp. 813–818 (2016)
15. Johnston, S.: RUP Plug-In for SOA V1.0. IBM developerWorks (2005)
16. Kazanavičius, J., Mažeika, D.: Migrating legacy software to microservices architecture. In: 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream),pp. 1–5 (2019)
17. Krämer, M.: A microservice architecture for the processing of large geospatial data in the cloud. Dissertation, Technische Universität Darmstadt, Darmstadt (2017)

18. Lewis, J., Fowler, M.: Microservices: a definition of this new architectural term (2014). https://martinfowler.com/articles/microservices.html
19. Marz, N., Warren, J.: Big Data: Principles and Best Practices of Scalable Real-time Data Systems, 1. aufl. edn. Manning, Shelter Island (2015)
20. Mayring, P.: Qualitative content analysis: theoretical foundation, basic procedures and software solution. Klagenfurt (2014)
21. Mazlami, G., Cito, J., Leitner, P.: Extraction of microservices from monolithic software architectures. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 524–531 (2017)
22. Papazoglou, M.: Web Services: Principles and Technology. Pearson, Prentice Hall, Upper Saddle River (2008)
23. Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: an update. Inf. Softw. Technol. **64**, 1–18 (2015)
24. Richardson, C.: Microservice Patterns: With Examples in Java. Manning, Shelter Island (2019)
25. Starke, G.: Effektive Softwarearchitekturen: Ein praktischer Leitfaden, 7 edn. Hanser eLibrary, Hanser, München (2015)
26. Taibi, D., Lenarduzzi, V., Pahl, C.: Architectural patterns for microservices: a systematic mapping study. In: Proceedings of the 8th International Conference on Cloud Computing and Services Science, vol. 1, pp. 221–232 (2018)
27. Wilde, T., Hess, T.: Forschungsmethoden der Wirtschaftsinformatik: Eine empirische Untersuchung. WIRTSCHAFTSINFORMATIK **49**(4), 280–287 (2007)
28. Zimmermann, O.: An architectural decision modeling framework for service-oriented architecture design. Dissertation@Stuttgart, University (2009)