An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A systematic literature review

Article · February 2019

CITATIONS
8

READS
1,125

3 authors:

Felipe Osses
Universidad Técnica Federico Santa María
5 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Gastón Márquez
Universidad Técnica Federico Santa María (sede Concepción)
41 PUBLICATIONS **262** CITATIONS
SEE PROFILE

Hernán Astudillo
Universidad Técnica Federico Santa María
188 PUBLICATIONS **1,247** CITATIONS
SEE PROFILE

An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A systematic literature review

Felipe Osses, Gastón Márquez, and Hernán Astudillo

Toeska research group
Universidad Técnica Federico Santa María
Valparaíso, Chile
felipe.osses@sansano.usm.cl, gaston.marquez@sansano.usm.cl,
hernan@inf.utfsm.cl

Abstract. Microservices are becoming a leading architectural choice in the service-oriented software industry. This approach proposes to develop applications as a suite of small services, each running on its own process and intercommunicating with lightweight mechanisms. Currently, there is still no clear perspective of emerging recurrent solutions (architectural patterns) or design decisions (architectural tactics) that influence microservices. This article describes an exploratory study, based on a systematic review, of academic literature related to architectural patterns and tactics proposed for microservices. The review yield 1067 initial studies, which reported: 44 architectural patterns of microservices; architectural tactics related to microservices dependent on other disciplines; and it was also found that most of architectural patterns and tactics are associated to five quality attributes: scalability, flexibility, testability, performance, and elasticity. Added to that, it was noticed that most microservices are reported in articles related to DevOps and IoT. Finally, a new proposal of microservices pattern taxonomy is suggested.

Keywords: Microservices, architectural patterns, architectural tactics, taxonomy, systematic literature review

1 Introduction

Service-oriented architecture (SOA) has emerged as a medium of developing distributed systems based on components as services. Along with SOA, cloud computing is gaining popularity among mid-size and small business allowing the emergence of new trends in software engineering, such as *microservices*. A microservice architectural style is an approach to developing a single application as a suite of small services, each running in its process and communicating with lightweight mechanisms. These services are built around business capabilities and independently deployable by fully automated deployment machinery [16].

Although microservices have emerged from industry [18], academic research has put the focus of interest in this approach. To understand microservices,

some studies have been conceived, like [10] [24] [3] [11]. These studies reveal that there is a great interest and emerging trends in microservices, but they do not describe those recurring solutions (*architectural patterns*) or design decisions (*architectural tactics*) that are emerging in this approach. Architectural patterns describe an abstract high-level system structure and its associated behavior, and a architectural tactic is a design decision that influences the achievement of system requirements [6]. According to [6], architectural tactics are "architectural building blocks" from which architectural patterns are created. Therefore, the objective of our study is to explore what architectural patterns and tactics have been described through a systematic review of the literature limited to academic papers to know what kind of proposals have been proposed.

The rest of the article is structured as follows: in Section 2 we describe the motivation and we illustrate those contributions related to systematic reviews that have been accomplishing for microservices. Subsequently, in Section 3 we mention the planning of the study that we have done with the objective of describing in Section 4 the results and the analysis obtained. In Section 5 we detail our proposal of taxonomy for microservices architectural patterns. In Section 6 we outline the threats to the validity of our research, and finally in Section 7, we detail the conclusions of our study.

2 Motivation and related work

The microservice architectural style is becoming a preferred industrial approach to develop applications as suites of small services, each running in its process and communicating with lightweight mechanisms [18] [16]. One of the *avant-garde* companies in adopting the microservice architecture approach is Netflix [1] handling about two billion API edge requests every day composed of approximately 500 microservices. On the other hand, Uber has about 1300 microservices to improve reliability and scalability [2].

There is an interesting set of reviews in the field of microservices. Di Francesco et al. [10] contributes to the study of microservices with a reusable framework for classifying, comparing, and evaluating architectural solutions, methods, and techniques for microservices. Proposals like [24], [3], [19], and [11] not only analyze the emerging standards in microservices but also the types of research conducted and the practical motivations around carrying out the microservices architecture. There are other proposals for microservices architectural patterns in industry such as patterns language [21] and others; but we have focused on academic papers for this review. The previous works show the current evidence on microservices, but none of them describe the architectural patterns and tactics proposed for this discipline. Therefore, this gives us the motivation to perform this systematic review.

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

[2] http://www.theserverside.com/feature/How-microservices-patterns-helped-Uber-systems-perform-better

3 Review planning

In this section, we will describe the research method executed in this study. This is based on procedures for performing systematic reviews of Barbara Kitchenham [14], who propose appropriate guidelines for systematic reviews for software engineering researchers [15]. The research planning is defined in Figure 1. To reduce the possibility of bias, the review was developed by three researchers. The execution of the protocol was done between June 1st and November 30th.

3.1 Research questions

To establish the research questions, we set up meetings with experts using the brainstorming technique[3] focused on the next keywords: "software architecture", "microservices", "quality attributes", "patterns" and "tactics". In those meetings we establish the next research questions:

RQ1: *What architectural patterns have been proposed for microservices?*

RQ2: *What architectural tactics have been proposed for microservices?*

RQ3: *What are the quality attributes associated with architectural patterns and tactics in microservices?*

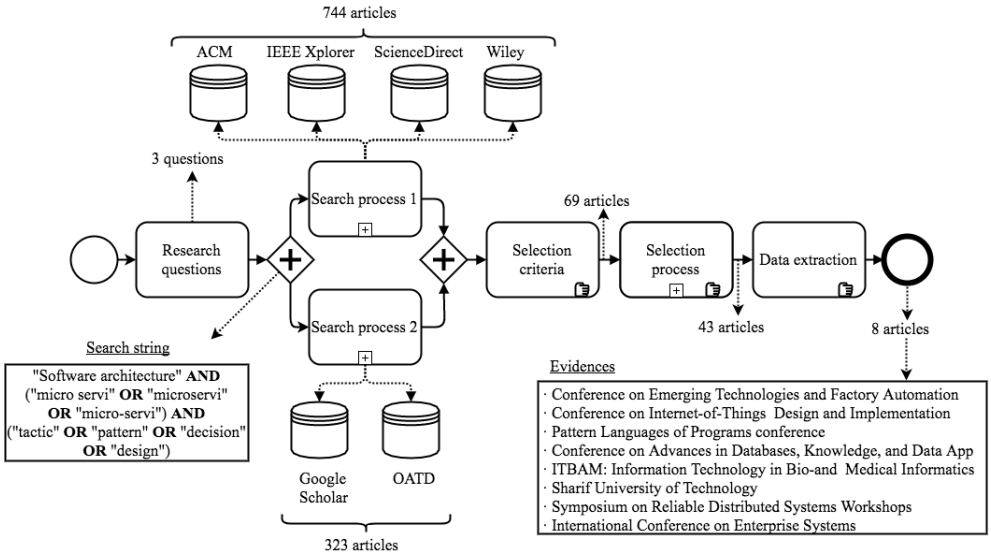

Fig. 1. Planning overview

[3] https://www.mindtools.com/brainstm.html

3.2 Search process

In this process, we establish the search string related to the research questions. Therefore, we use the following search terms to find relevant studies, using and combining all the terms by the boolean AND/OR operators. In the search strings, there are two-word bases: *software architecture* and *microservices*, which should be included in all the questions. Thus, it was decided not only use "tactic" and "patterns" words but also we decided to expand the search string using the words "decision" and "design." For microservices, we included in the string the words "micro servi", "microservi" or "micro-servi" (see Figure 1).

The following step on the search process is to establish where the search would be performed. In this study was decided to concentrate the efforts of search on academic studies on the most important databases of journals and conferences in the area of software engineering. 744 papers were obtained using the next databases (Search Process 1): ACM Digital Library[4], IEEE Xplore[5], ScienceDirect-Elsevier[6], and Wiley InterScience[7]. However, due to the novelty of the microservices and the intention to expand the search and increase the knowledge, it was decided to performed a second search in the following databases, obtaining 323 new papers (Search Process 2): Google Scholar[8], and OATD - Open Access Thesis Databases[9]. In total, 1067 papers were obtained from digital libraries.

3.3 Selection criteria

Each study was analyzed using inclusion and exclusion criteria, in order to verify whether paper is suitable to respond the research questions. The inclusion and exclusion criteria used in this study are described as follows:

Inclusion Criteria:(1) Evidence related to software engineering and software architecture, (2) all evidence published in journals and conferences proceedings in the field of software architecture and software engineering, and (3) all evidence in English.

Exclusion Criteria: (1) Gray literature (e.g. books, white papers, etc.), (2) studies which address only commercial-off-the-shelf (COTS), (3) when the same study is reported by more than one paper, only consider the most complete study, (3) evidence that do not explicitly relate to microservices, architectural patterns or tactics or not relate to the research questions, and (4) evidence that are not in English.

---

[4] http://portal.acm.org
[5] http://www.ieee.org/web/publications/xplore/
[6] http://www.elsevier.com
[7] http://www3.interscience.wiley.com
[8] https://scholar.google.com/
[9] https://oatd.org/

3.4 Selection Process and Data extraction

The selection of studies was performed based on the next sequence: (a) The first step was to search and identify relevant studies in the selected databases using the search terms, (b) after obtaining the relevant studies, the inclusion and exclusion criteria were applied, (c) the third step was manually excluded studies based on the title, abstracts, introduction, and conclusion of the papers, and (d) finally, we obtain the last list of papers.

Once we apply the selection criteria and process, we use a spreadsheets to extract relevant data used (you can consult the sheet in the next link[10]). For each selected paper, we fill the next fields: paper ID, publication year, paper title, authors, type of study (conference, journal, etc.), and keywords. At the time of manually analyzing each paper, we realized that the vast majority of papers did not offer the complete information to answer our RQs. Therefore, we analyze each paper in depth and discard those that did not meet our expectations. This manual process involved a considerable reduction of the evidence.

4 Results and analysis

We found academic literature evidence in several events (see Figure 1). ICSA (International Conference on Software Architecture) and ECSA (European Conference on Software Architecture) were considered in the first phases of our revision, but we didn't find evidence on architectural tactics and pattern in microservices in these conferences. To be sure of this lack of evidence, we review the papers related to these conferences in IEEExplore and ACM digital library manually (from 2010 to 2017). Then, we will proceed to answer the research questions created for this study.

RQ1: What architectural patterns have been proposed for microservices? Table 1 summarize the 44 architectural patterns of microservices found in our review. We have decided to summarize them using as reference the schema proposed in [2], which are: reference, ID (MSP, MicroService Pattern), name, context, problem and solution.

Although the patterns evidenced in this review are directly or indirectly related to microservices, we will describe those patterns that we have verified that belong directly. The selection of these patterns is based on their contribution to the development of microservice architecture.

- Name: **Microservices Architecture**
  - *Context*: Is required an modular application, and the modules should be independent.
  - *Problem*: How do the architect an application as a set of independent modules?

[10] https://drive.google.com/open?id=1sxbPsYxe8ZtptP7vZKjG5XRUEr03LA6hCJmR6S1ADnk

Table 1. Architectural patterns in microservices in academy

Ref.	ID	Name
[7]	**AMSP1**	*Modern Web Architecture*
	AMSP2	*Single Page Application*
	AMSP3	*Native Mobile Application*
	AMSP4	*Near Cache*
	AMSP5	*Microservices Architecture*
	AMSP6	*Business Microservice*
	AMSP7	*Backend for Frontend*
	AMSP8	*Adapter Microservice*
	AMSP9	*Results Cache*
	AMSP10	*Page Cache*
	AMSP11	*Scalable Store*
	AMSP12	*Key Value Store*
	AMSP13	*Document Store*
	AMSP14	*Microservices DevOps*
	AMSP15	*Log Aggregator*
	AMSP16	*Correlation ID*
	AMSP17	*Service Registry*
[4]	**AMSP18**	*Enable Continuous Integra.*
	AMSP19	*Recover the Current Arch.*
	AMSP20	*Decompose the Monolith*
	AMSP21	*Decom. based on data*
	AMSP22	*Change code depen.*
[4]	**AMSP23**	*Service discovery*
	AMSP24	*service discovery client*
	AMSP25	*Internal load balancer*
	AMSP26	*external load balancer*
	AMSP27	*Intro. circuit breaker*
	AMSP28	*Intro. config. server*
	AMSP29	*Intro. edge server*
	AMSP30	*Containerize the Services*
	AMSP31	*Deploy Cluster*
	AMSP32	*Monitor the system*
[8]	**AMSP33**	*Self-containment of services*
	AMSP34	*Circuit Breaker*
	AMSP35	*Load Balancer*
	AMSP36	*Container*
	AMSP37	*Handling diff. service*
	AMSP38	*Blue green deployment*
	AMSP39	*Canary release*
[17]	**AMSP40**	*Database is the service*
[20]	**AMSP41**	*IoT edge provisioning*
	AMSP42	*IoT edge code deployment*
	AMSP43	*IoT edge orchestration*
	AMSP44	*IoT edge diameter of things*

  - *Solution*: Design the application with a set of modules where each is an independent business function with its own data, developed by a separate team and deployed in a separate process.
- Name: **Backend for Frontend**
  - *Context*: The Business Microservices, that encapsulate individual business functions, do not map cleanly to the channel-specific needs of the client applications.
  - *Problem*: How to represent a channel-specific service interface that is consistent with an overall microservices architecture but allows enough uniqueness that it can be adapted to the needs of a specific client type?
  - *Solution*: Build a Backend for Frontend (BFF) that acts as a single API for a client. Implement differents BFFs for different types of clients, each with an API customized to what that client type needs.
- Name: **Microservices DevOps**
  - *Context*: Is neccesary to put a system into production.
  - *Problem*: How to balance the needs of developers (which want to work on small, easy-tomodify artifacts) and operations teams, which need to minimize the impact of changes on critical systems in order to maintain availability?
  - *Solution*: Isolate each microservice as much as possible, while being able to easily and quickly identify and resolve issues with the microservices.

Applying unique CI/CD pipelines to each microservice will allow you to build and deploy each microservice individually.

- Name: **Service Registry**
  - *Context*: Many different business microservices comprising an application.
  - *Problem*: How to decouple the physical address (IP address) of a microservice instance from its clients so that the client code will not have to change if the address of the service changes?
  - *Solution*: Map between a unique identifier and the current address of a service instance in order to decouple the physical address of a service from the identifier.
- Name: **Change Code Dependency to Service Call**
  - *Context*: A software system has been decomposed to a set of small services to use microservices architectural style. There is a component in the system that is acting as a dependency to another services or components.
  - *Problem*: When it is appropriate to change this code-level dependency to service-level dependency? And when it is not?
  - *Solution*: Try to keep the services code as separate as possible. When services shares code as a dependency there is a high chance that a service developer can fail the build process of another service by changing the shared code.
- Name: **Deploy into a Cluster and Orchestrate Containers**
  - *Context*: A software system has been architected based on microservices architectural style, and a continuous integration pipeline is in place and is working.
  - *Problem*: How to deploy a service's instances into a cluster? How to orchestrate the deployment and re-deployment of all of the services with the least effort?
  - *Solution*: There should exists a system that can manage a cluster of computing nodes. This management system should be able to deploy the services container images on-demand, with specified number of instances and on different nodes. Additionally, it should handle the failure of instances and restart the failed nodes or instances in case.
- Name: **Container**
  - *Context*: Individual services can be hosted as a single container.
  - *Problem*: How to simplify deploying applications?
  - *Solution*: The use of containers enclose the microservice itself, including all required libraries and data, which also supports the requirement of self-containment. This in turn provides several advantages: better testability, ease of service deployment better scalability.
- Name: **Database is the Service**
  - *Context*: The addition of new behaviors and business logic at the database level may be a possible approach to reduce complexity, and thus the related risks, and also to gain improvements in terms of speed and scalability.

- *Problem*: If each scalable service has its own database (cluster), is there any way to reduce the complexity of the architecture and the related risks, while also gaining more improvements in terms of speed and scalability?
- *Solution*: Create a service whose sole responsibility will be to keep the databases in sync.

– Name: **REST Integration**

- *Context*: Intuitive way of integrating microservices
- *Problem*: How to synchronize information between microservices?
- *Solution*: Dedicated REST endpoint that other services can use to notify them.

Given the volume of architectural patterns in microservices found in this review, we performed a quality assessment with the objective to illustrate those architectural patterns that, under our criteria, satisfy the following four quality criteria (QC) questions. Each QCs was evaluated using a adaptation of the York University, Centre for Reviews and Dissemination (CDR) Database of Abstracts of Reviews of Effects (DARE) criteria [1]. Some contributions of Kitchenham et al. [15] [13] recommend applying this type of evaluations with the objective of appraise their quality and highlight strengths and weaknesses of the contributions (in our case, the architectural patterns).

QC1: The architectural pattern describes the *context* in which it is applied? Y (yes), the architectural pattern clearly describes the context in which it is located; P (partly), the architectural pattern partially describes the context in which it is located; and N (no), the architectural pattern vaguely describes the context.

QC2: The architectural pattern describes the *problem* it is trying to solve? Y, the architectural pattern openly describes the problem it is trying to solve; P, the architectural pattern does not directly describe the problem to be solved; N, Unable to visualize which problem the architectural pattern is trying to solve.

QC3: The architectural pattern describes the *solution* regarding the context and problem? Y, the architectural pattern clearly describes the solution proposed; P, the architectural pattern partially describes the proposed solution; N, It can not be described what solution the architectural pattern proposes.

QC4: Is the architectural pattern related to one or more quality attributes? Y, the architectural pattern allows one or more quality attributes to be identified in the description; P, the architectural pattern allows identifying at most one attribute of quality in the description; N, it is not possible to identify attributes of quality in the pattern.

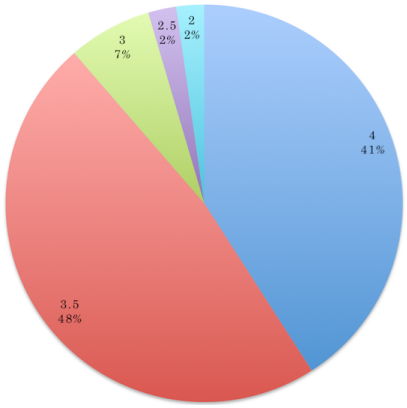

Fig. 2. Total scores distribution

Table 2. Quality evaluation of MSPs

ID	QC1	QC2	QC3	QC4	Total score	ID	QC1	QC2	QC3	QC4	Total score
MSP1	Y	Y	Y	P	3.5	MSP23	Y	Y	Y	P	3.5
MSP2	P	Y	Y	Y	3.5	MSP24	Y	Y	Y	P	3.5
MSP3	Y	Y	Y	Y	4	MSP25	Y	Y	Y	Y	4
MSP4	Y	Y	Y	P	3.5	MSP26	Y	Y	Y	Y	4
MSP5	Y	Y	Y	P	3.5	MSP27	Y	Y	Y	P	3.5
MSP6	Y	Y	Y	Y	4	MSP28	Y	Y	Y	P	3.5
MSP7	Y	Y	Y	Y	4	MSP29	Y	Y	Y	P	3.5
MSP8	Y	Y	Y	P	3.5	MSP30	Y	Y	Y	Y	4
MSP9	Y	Y	Y	P	3.5	MSP31	Y	Y	Y	Y	4
MSP10	Y	Y	Y	P	3.5	MSP32	Y	Y	Y	P	3.5
MSP11	Y	Y	Y	Y	4	MSP33	Y	P	Y	P	3
MSP12	P	Y	Y	Y	3.5	MSP34	Y	P	Y	Y	3.5
MSP13	Y	Y	Y	Y	4	MSP35	N	P	Y	Y	2.5
MSP14	Y	Y	Y	P	3.5	MSP36	Y	N	Y	Y	3
MSP15	Y	Y	Y	P	3.5	MSP37	Y	P	N	P	2
MSP16	Y	Y	Y	Y	4	MSP38	Y	P	Y	Y	3.5
MSP17	Y	Y	Y	Y	4	MSP39	P	P	Y	Y	3
MSP18	Y	Y	Y	Y	4	MSP40	Y	Y	Y	Y	4
MSP19	Y	Y	Y	P	3.5	MSP41	Y	Y	Y	P	3.5
MSP20	Y	Y	Y	Y	4	MSP42	Y	Y	Y	Y	4
MSP21	Y	Y	Y	Y	4	MSP43	Y	Y	Y	P	3.5
MSP22	Y	Y	Y	Y	4	MSP44	Y	Y	Y	Y	4

The scoring procedure was $Y = 1$, $P = 0.5$, $N = 0$. Table 2 describe the final results of the quality evaluation of the architectural patterns. The trend shows that a total of 18 of 44 architectural patterns satisfy our quality criterias, which is equivalent to 48% of the total (See Figure 2). This gives us to understand that

almost half of the architectural academic patterns have a complete description. However, those patterns with low score (between 2 and 2.5) that are equivalent to 4% of the MSPs call our attention. These architectural patterns are **MSP35** and **MSP37**. Although the authors of these architectural patterns describe their potential, it does not satisfy our quality criteria since they do not clearly show their contributions.

Key finding (1): *The academy is joining microservices architectural patterns to other disciplines, for example DevOps and IoT*

RQ2: What architectural tactics have been proposed for microservices? The architectural tactics found in this review are not directly related to microservices. We note that architecture tactics for microservices as such do not exist in academia. However, there are architectural tactics that complement each other with other disciplines, such as DevOps (Development and Operations) [9] [5].

Another article found in the review dealing with architectural tactics indirectly is [26]. In the investigation, they describe two indications to what may be architectural tactics: (1) *scaling* and (2) *independence*. From one side, (1) detail that a microservice must be independently deployable with each microservice having its deployment architecture; and not share its resources like containers, caches, datastores with other enterprise software components. On the other side, (2) is complemented by (1) and detail that scaling of one microservice is independent of other microservices; and increasing the capacity of a microservice, by moving its hosted target or by increasing the number of instances, should not have any impact on its consumers. Both (1) and (2) are not architectural tactics directly. However, the authors refer to these words as decisions in microservice architecture. Therefore, we consider that they may be potential of unexplored architectural tactics.

Key finding (2): *In the academic field, not exist sufficient evidence on architectural tactics for microservices*

RQ3: What are the quality attributes associated with architectural patterns and tactics in microservices? Since on the architectural tactics side we do not find enough evidence, we focus our efforts on architectural patterns. To rescue the quality attributes (QA) associated with the 44 architectural patterns found in our studies, we used information retrieval techniques, such as the Rapid Automatic Keyword Extraction (RAKE) algorithm [22]. After apply some manual filters from the keywords obtained by the algorithm, the QAs with the highest score were: scalability, flexibility, testability, elasticity, and performance.

The most prevalent QAs are are scalability and performance. For scalability, the architectural patterns reveal that exists demand and a need for scalability with microservices, just as there are with a traditional IT architecture and software systems deployment. On the performance side, it is clear that this QA is related to scalability. But, despite the obvious importance of a sufficient level

of performance, there is still a lack of performance engineering approaches and architectural patterns explicitly taking into account the particularities of microservices [12].

Key finding (3): *Scalability and performance are trends in microservices architectural patterns*

5 Microservices architectural pattern taxonomy

The contribution of our taxonomy is based on naming, describing and classifying the architectural patterns for microservices find in this revision. To create our taxonomy, we follow the strategy proposed by Velardi et al. [23] which is an automated procedure to achieve a significant speed-up factors (in our case, categories) in the development of resources (architectural patterns) with human validation and refinement. Next, we will proceed to describe the classification for architectural patterns in microservices (see Figure 3):

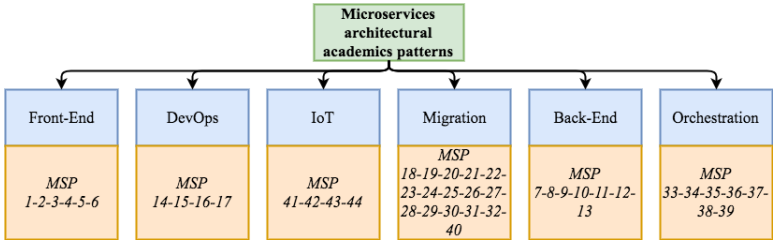

Fig. 3. Proposed taxonomy

IoT patterns: The descriptions of the patterns reveal that microservices and virtualization have changed the industry bringing agility and innovation in this domain.

DevOps patterns: Microservice architecture emerged from a standard set of DevOps ideologies. The evolution of both microservices and DevOps was not just limited to transforming monolithic applications into decomposed services.

Front-End patterns: Our revision reveals that all front-end functionalities are part of a single application and communicate with the back-end that is split into microservices.

Back-End patterns: This category is closely related to the Front-End category. This category describes the layer between the user experience and the resources it calls on.

Orchestration patterns: The orchestration is a concept inherited from SOA. When a service-oriented architecture uses an orchestration pattern for communication, there are point-to-point connections between the services.

Migration patterns: According to [4], the difficulties around the decomposition of a system into small services, and monitoring and managing these services are among the other important factors that make migrating to microservices a non-trivial and cumbersome task.

The taxonomy allows us to visualize those tendencies that academy describe when illustrating recurring solutions in the discipline of microservices.This information gives a practitioner an array of tried and tested solutions to common problems, thus reducing the technical risk to the project by not having to employ a new and untested design.

6 Threats to validity

In this section, we will as a reference the threat list defined in [25] to conduct studies. To minimize the threats to internal validity, the research team that performed the revision discussed each article found based on a scheme, which has the following structure: paper ID, full paper summary, introduction summary, description of the proposal, brief description of the sections of the article, and observations and comments. Some cases, such as the **MSP30** and **MSP36** patterns have a lot of similarity in the name. However, the authors define architectural patterns differently, so we must to include them. On the side of external validity, we have discarded those works that have not been able to validate concretely both the architectural patterns and tactics. In that sense, we were rigorous, because the scientific community is categorical in requesting evidence of the proposals described in the articles. Also, we excluded the so-called *grey literature* (e.g., white papers, editorials, etc.); nevertheless, this issue did not affect our study significantly since we considered papers have undergone a rigorous review process.

Finally, to reduce the selection bias, three researchers made the selection of papers during our revision. Two researchers applied the inclusion and exclusion criteria, compared results and resolved conflicts following the definitions stated in the research protocol. A third person was in charge of managing the selection and participating whenever a resolution is hard to make by the other two. The same three researchers followed the same inclusion/exclusion criteria. Therefore, there was no new confounding factor in the selection of relevant sources from either the included journals/conferences.

7 Conclusions

This article exhibits a summary of architectural patterns and tactics proposed for microservices. To perform the exploratory study, a systematic literature review was performed revealing 8 significant papers. For architectural patterns, the review reported 44 architectural patterns for microservices allowing us to propose a taxonomy based on them. This taxonomy provides a categorization of the 44 architecture patterns for microservices based in Front-End patterns, Back-End patterns, Orchestration patterns, Migration Patterns, IoT patterns and DevOps

patterns. On the side of architectural tactics, we did not find tactics related to microservices, allowing us to know and understand that it is an unexplored area. About quality attributes, we found that most of architectural patterns and tactics are associated with five quality attributes: flexibility, testability, elasticity, performance and scalability, the latter two being the most dominant of them.

For researchers, the analysis reveal that there are still significant challenges and issues for future research in the area of architectural patterns/tactics and quality attributes related to microservices. For practitioners, future work is expected to tighten ties between microservices, DevOps and IoT.

Acknowledgments

This work has been financially supported by (1) Comisión Nacional de Investigación Científica (CONICYT) CONICYT-PCHA/Doctorado Nacional/2016-21161005 and (2) FONDECYT regular 1140408. We also appreciate the support of the Chilean Navy.

References

1. Effectiveness matters. the database of abstracts of reviews of effects (dare). United Kingdom: The University of York. https://www.york.ac.uk/media/crd/em62.pdf (2002)
2. Alexander, C., Ishikawa, S., Silverstein, M., i Rami, J.R., Jacobson, M., Fiksdahl-King, I.: A pattern language. University of California, Berkeley (1977)
3. Alshuqayran, N., Ali, N., Evans, R.: A systematic mapping study in microservice architecture. IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA) pp. 44–51 (2016)
4. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Microservices migration patterns. Technical Report No. 1, Automated Software Engineering Group, Sharif University of Technology (2015)
5. Bass, L., Weber, I., Zhu, L.: Devops: A software architect's perspective. Addison-Wesley Professional (2015)
6. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice (3rd Edition). SEI Series in Software Engineering (2013)
7. Brown, K., Woolf, B.: Implementation patterns of microservices architectures. Conference on Pattern Languages of Programs (PLoP) (2016)
8. Butzin, B., Golatowski, F., Timmermann, D.: Microservices approach for the internet of things. IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA) pp. 1–6 (2016)
9. Chen, H.M., Kazman, R., Haziyev, S., Kropov, V., Chtchourov, D.: Architectural support for devops in a neo-metropolis bdaas platform. IEEE 34th Symposium on Reliable Distributed Systems Workshop (SRDSW) pp. 25–30 (2015)
10. Di Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: Trends, focus, and potential for industrial adoption. IEEE International Conference on Software Architecture (ICSA) pp. 21–30 (2017)
11. Dragoni, N., Giallorenzo, S., Lluch-Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow. CoRR (2016)

12. Heinrich, R., van Hoorn, A., Knoche, H., Li, F., Lwakatare, L.E., Pahl, C., S., S., Wettinger, J.: Performance engineering for microservices: Research challenges and directions. Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion pp. 223–226 (2017)
13. Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., Linkman, S.: Systematic literature reviews in software engineering – A systematic literature review. Information and Software Technology 51(1), 7–15 (2009)
14. Kitchenham, B.: Procedures for performing systematic reviews. Tech. rep., Keele University (2004)
15. Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Engineering 2, 1051 (2007)
16. Lewis, J., Fowler, M.: Microservices-a definition of this new architectural term. https://martinfowler.com/articles/microservices.html (2014)
17. Messina, A., Rizzo, R., Storniolo, P., Urso, A.: A simplified database pattern for the microservice architecture. The Eighth International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA) (2016)
18. Newman, S.: Building microservices: designing fine-grained systems. O'Reilly Media, Inc. (2015)
19. Pahl, C., Jamshidi, P.: Microservices: A systematic mapping study. Proceedings of the 6th International Conference on Cloud Computing and Services Science CLOSER 1, 137–146 (2016)
20. Qanbari, S., Pezeshki, S., Raisi, R., Mahdizadeh, S., Rahimzadeh, R., Behinaein, N., Mahmoudi, F., Ayoubzadeh, S., Fazlali, P., Roshani, K., Yaghini, A., Amiri, M., Farivarmoheb, A., Zamani, A., Dustdar, S.: Iot design patterns: Computational constructs to design, build and engineer edge applications. IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI) pp. 277–282 (2016)
21. Richardson, C.: A pattern language for microservices. http://microservices.io/patterns/ (2017)
22. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic keyword extraction from individual documents. Text Mining: Applications and Theory pp. 1–20 (2010)
23. Velardi, P., Cucchiarelli, A., Petit, M.: A taxonomy learning method and its application to characterize a scientific web community. IEEE Transactions on Knowledge and Data Engineering 19(2) (2007)
24. Vural, H., K.M., Guney, S.: A systematic literature review on microservices. International Conference on Computational Science and Its Applications pp. 203–217 (2017)
25. Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A.: Experimentation in software engineering. Springer Science Business Media (2012)
26. Xiao, Z., Wijegunaratne, I., Qiang, X.: Reflections on soa and microservices. 4th International Conference on Enterprise Systems (ES) pp. 60–67 (2016)