Metadata of the chapter that will be visualized online

Chapter Title	Strategies Reported in the Literature to Migrate to Microservices Based Architecture	
Copyright Year	2019	
Copyright Holder	Springer Nature Switzerland AG	
Author	Family Name	**Silva Filho**
	Particle	
	Given Name	**Heleno Cardoso da**
	Suffix	
	Address	Salvador, Brazil
	Email	heleno.filho@area1.edu.br
Corresponding Author	Family Name	**Figueiredo Carneiro**
	Particle	
	Given Name	**Glauco de**
	Suffix	
	Organization	Universidade Salvador (UNIFACS)
	Address	Salvador, Brazil
	Email	glauco.carneiro@unifacs.br
Abstract	Context: Microservice-oriented architecture relies on the implementation and deployment of small and autonomous microservices, rather than implementing the functionalities in one unique module to be deployed. They have been adopted as a solution to the shortcomings of the monolithic architecture such as lack of flexibility. Goal: This paper discusses lessons learned and challenges reported in the literature regarding the migration of legacy monolithic software systems to microservices based architecture. Method: We performed an automated search targeting public repositories to accomplish the stated goal. Results: Based on the evidence provided by 12 studies, we classified main findings in lessons learned related to the migration, as well as associated difficulties and challenges. Conclusions: the guidelines to migrate to microservices based architecture are maturing/evolving and the literature has pinpointed issues that deserve further investigation.	
Keywords (separated by "-")	Legacy software systems - Monolithic - Microservices - Cloud computing	

81 Strategies Reported in the Literature to Migrate to Microservices Based Architecture

Heleno Cardoso da Silva Filho and Glauco de Figueiredo Carneiro

81.1 Introduction

AQ2 Microservices are a suite of usually small autonomous deployable and modular services running an unique process. They are network-accessible and communicate through well-defined, lightweight mechanisms to serve a business goal. Microservices can be effective to build complex software solutions in less time when compared to traditional software architectural solutions [1]. In fact, not only microservices but also the container-based approaches are associated with the boom of the so called cloud-native applications [2]. Microservices are a promising target to encourage the modernization of monolithic legacy applications to allow to take advantage of the benefits provided by cloud computing [3].

The aim of this work is to provide an updated overview regarding the lessons learned reported in the literature related to the migration and associated difficulties/challenges of this process. The migration from a monolithic architecture into microservices is not trivial due to decisions such as how to distribute the legacy functionalities into microservices and establish the dependencies among them in order to preserve their originality [4].

The remainder of this paper is organized as follows. The background and related work are presented in Sect. 81.2. Section 81.3 describes the steps we followed to find relevant studies in the literature. In Sect. 81.4, we discuss lessons learned, difficulties, benefits and challenges of the migration process based on evidence obtained from the selected studies. Finally, in Sect. 81.5, we present conclusions and opportunities for future research.

81.2 Background

The migration to a microservice based architecture relies on the principle that the main focus must be on the services instead of infrastructure owner-ship [5]. Microservice, the software equivalent of toy bricks, have been enjoyed increasing popularity and diffusion in industrial environments to build complex solutions [1, 6]. The microservice architecture is the result of applying the single responsibility principle at the architectural level [4]. The high level of independence of microservices allows them to be separately deployable from each other. This enables that parts of the application can be changed and updated without affecting other parts [7]. Considering that the number of services involved in applications based on the microservices architecture can increase, manual deployment processes is no more effective in such architectures due to the frequency of new deployments. For this reason, automated deployment solutions such as continuous delivery pipelines can be an effective solution for this situation [7]. As a result of the services independence, the underlying technologies and adopted programming languages can be diversified. If one service can be implemented using Java EE, another one could be implemented using .Net, Ruby, or Node.js [7]. Microservice architectures illustrate the principle of *smart endpoints* and *dumb pipes* [8], where the lightweight and minimal middleware components such as messaging systems are *the dumb pipes* and the intelligence of each service is *the smart endpoint* [7].

H. C. da Silva Filho
Salvador, Brazil
e-mail: heleno.filho@area1.edu.br

G. de Figueiredo Carneiro (✉)
AQ1 Universidade Salvador (UNIFACS), Salvador, Brazil
e-mail: glauco.carneiro@unifacs.br

© Springer Nature Switzerland AG 2019
S. Latifi (ed.), *16th International Conference on Information Technology-New Generations (ITNG 2019)*,
Advances in Intelligent Systems and Computing 800,
https://doi.org/10.1007/978-3-030-14070-0_81

81.3 Steps to Find Relevant Studies in the Literature

This section describes the steps to find relevant studies in the literature. We do not intend to perform a precise and rigorous literature review, based on well-defined and structured review protocols to extract, analyze, and document results [9]. Our goal is to perform a non-structured review process to gather information to answer the research questions of this paper as follows: **Research Question 1 (RQ1)**: *Which strategies have been reported in the literature to support the migration of legacy software systems to microservices-based architecture?* The knowledge of strategies applied both by researchers in the academia and practitioners in the industry to support the migration of legacy systems to microservices can be an opportunity to encourage them to embrace this challenge. **Research Question 2 (RQ2)**: *Which lessons learned have been reported in the literature regarding challenges and advantages perceived as a consequence of the aforementioned migration?* The reported challenges and advantages are key to improve mentioned strategies for the migration.

To answer the research questions, we performed a search in ACM Digital Library, Science Directory, IEEE Xplore and Springer scientific repositories to identify relevant studies published in top software engineering venues. A selection of keywords was made to perform the search in the mentioned repositories. We used an adjusted version of the PICOC method proposed by Petticrew and Roberts [10] as presented in Table 81.1.

We defined the following search string to identify the primary studies in the target scientific repositories to select studies published from 2008 to 2018: *((software or application or monolithic) and migration and microservice)*. Considering differences in the syntax of the target search engines, we adjusted the search string in each repository to fulfill their respective search requirements as presented in Table 81.2. All searches were conducted in May 4th, 2018. The result of the automated search returned a set of 95 studies (14 IEEE, 13 ACM, 25 SCD and 43 SPRINGER), from which we selected five studies that met the inclusion, exclusion and quality criteria as described bellow. Besides Moreover, we included seven studies based on suggestions of the authors (Tables 81.3, 81.4, and 81.5). AQ3

Table 81.1 Defining the search string

Component	Definition
Population (P)	Monolithic, software, application
Intervention (I)	Migration
Outcomes (O)	Microservices

Table 81.2 Adjusted search strings for each repository

Repository	Adjusted search strings
ACM Digital Library	"query": ((software or application) and monolithic and microservice) "filter": "publicationYear": "gte":2008, "lte":2018 , owners.owner=HOSTED, acmPubGroups.acmPubGroup=Journal & Proceeding
IEEE Xplore	((software or application) and monolithic and microservice) and refined by Year: 2008–2018
Springer	((software or application) and monolithic and microservice) and refined by Year: 2008-2018
Science Directory	2008 and ((software or application) and monolithic and microservice) [All Sources(Computer Science)]

Table 81.3 Inclusion criteria

Inclusion criteria	Criteria
IC1	The articles should address difficulties in migrating legacy systems to cloud-based architecture in the presence of cloud computing, software architecture, legacy system, migration, change, evolution, strategies, approaches, techniques, type of change, change category, support, analyze AND
IC2	The papers are reported in peer reviewed conference or Journal AND
IC3	The papers are reported in peer reviewed conference or Journal AND
IC4	The publication date of the article should be between 2008 and 2018

Table 81.4 Exclusion criteria

Exclusion criteria	Criteria
EC1	Articles that do not address difficulties in migrating legacy systems to cloud-based architecture in the presence of cloud computing, software architecture, legacy system, migration, change, evolution, strategies, approaches, techniques, type of change, change category, support, analyze OR
EC2	The papers are not published in a peer reviewed conference or journal OR
EC3	The papers are not described in English OR
EC4	Date of publication of the article outside the period 2008 e 2018 OR
EC5	Duplicated reports of the same study available in different sources, consider the most complete version of the study

81.4 Results and Discussion

We based on the results of the selection process presented in Sect. 81.3 to list and classify the selected primary studies in Table 81.6 aiming at enabling a clear understanding of the lessons learned and findings reported to answer research questions **RQ1** and **RQ2**. Results obtained from both the automatic string search in the selected repositories and manual inclusion were maintained in a dataset. Studies with the same title, author(s), year of publication and abstract were considered duplicated and thus discarded. We organized the selected studies based on fields as follows: (i) identification number; (ii) year; (iii) title; (iv) objectives or aims; (v) strategies to support the migration of legacy software systems to microservices-based architecture (**RQ1**); and (vi) lessons learned regarding challenges and advantages perceived as a consequence of the aforementioned migration (**RQ2**). In the following section, we discuss findings to answer **RQ1** and **RQ2**. These findings are presented in Fig. 81.1.

81.4.1 Evidence to Answer RQ1

According to Fig. 81.1, 12 studies (S05, S06, S08, S79, S88, S96, S97, S98, S99, S100, S101, S102) proposed different strategies to support the migration of legacy software systems to microservices-based architecture. These strategies are presented in the following paragraphs.

S05 proposed formal coupling strategies and the clustering algorithm to support the migration. The strategy consisted in transforming the monolith application into the graph representation, while the clustering step proposes a new version of the graph representation of the monolith into microservice candidates [11]. The authors also proposed a quality evaluation that can support software architects to execute the approach according to their specific needs, making viable the reduction of the team size and lowering the domain redundancy of extracted services [11].

S06 discussed the use of a dataflow-driven mechanism as a systematic methodology for microservice-oriented decomposition. It is an algorithm based on a semi-automated process intended to reduce the complexity during the decomposition practices [12].

S08 proposed a methodology to convert a monolithic system into an architecture based in microservices. The methodology consists in a sequence defined by the phases: analysis and design, implementation, testing and continuous integration within an evolutionary life cycle [13].

S79 proposed a solution based on the semantic similarity of functionalities related to the OpenAPI specifications. Through the use of a reference vocabulary, the approach seeks for potential candidates for microservices, as fine-grained groups of cohesive operations (and their associated resources) [14]. The approach has as input an OpenAPI specification of the application that describes its different interfaces, operations, and resources and Schema.org is given as reference vocabulary [14].

S88 discussed the use of relevant requirements for the decomposition of services through the *Service Cutter*, a knowledge management method and supporting tool framework for microservice decomposition that requires as input a set of specification documents and a set of weighted coupling criteria [15]. The output is a graph representing candidate microservices as nodes, and how cohesive and/or coupled two candidates are through the use of weighted arcs. The authors emphasized the intention to support the decision making process instead of automating it completely [15]. Despite been proposed for generic services applications, the framework proposed in [15] can be an useful support the migration from legacy software monolithic systems towards microservices.

In S96, the authors described a manual migration based on the identification of Domains, Non-functional Require-

Table 81.5 Quality criteria

Quality criteria	Criteria
QC1	Is the paper a primary study (or is it a review, secondary study or "lessons learned" document based on an expert point of view)?
QC2	Is there a clear statement of the goals of the research?
QC3	Is there an appropriate description of the context in which the research was performed?
QC4	Was the research design appropriate to address the aims of the research?
QC5	Was the data collected in a way that addressed the research issue?
QC6	Is there a clear statement of findings?

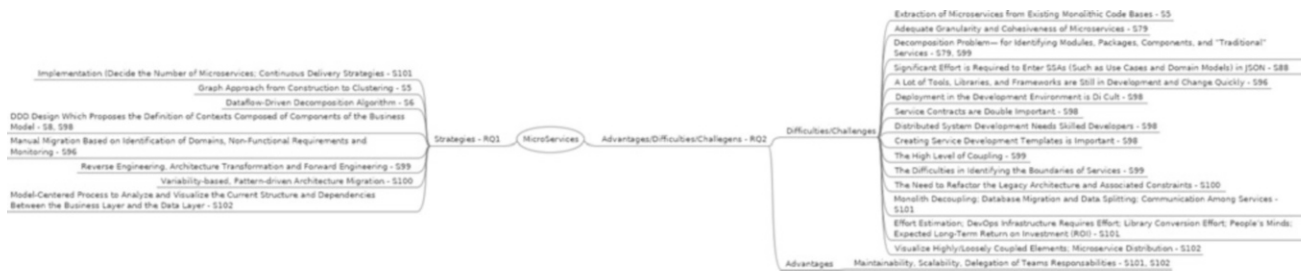
Implementation (Decide the Number of Microservices; Continuous Delivery Strategies - S101
Graph Approach from Construction to Clustering - S5
Dataflow-Driven Decomposition Algorithm - S6
DDD Design Which Proposes the Definition of Contexts Composed of Components of the Business Model - S8, S98
Manual Migration Based on Identification of Domains, Non-Functional Requirements and Monitoring - S96
Reverse Engineering, Architecture Transformation and Forward Engineering - S99
Variability-based, Pattern-driven Architecture Migration - S100
Model-Centered Process to Analyze and Visualize the Current Structure and Dependencies Between the Business Layer and the Data Layer - S102
Strategies - RQ1
MicroServices
Advantages/Difficulties/Challegens - RQ2
Difficulties/Challenges
Extraction of Microservices from Existing Monolithic Code Bases - S5
Adequate Granularity and Cohesiveness of Microservices - S79
Decomposition Problem— for Identifying Modules, Packages, Components, and "Traditional" Services - S79, S99
Significant Effort is Required to Enter SSAs (Such as Use Cases and Domain Models) in JSON - S88
A Lot of Tools, Libraries, and Frameworks are Still in Development and Change Quickly - S96
Deployment in the Development Environment is Di Cult - S98
Service Contracts are Double Important - S98
Distributed System Development Needs Skilled Developers - S98
Creating Service Development Templates is Important - S98
The High Level of Coupling - S99
The Difficulties in Identifying the Boundaries of Services - S99
The Need to Refactor the Legacy Architecture and Associated Constraints - S100
Monolith Decoupling; Database Migration and Data Splitting; Communication Among Services - S101
Effort Estimation; DevOps Infrastructure Requires Effort; Library Conversion Effort; People's Minds; Expected Long-Term Return on Investment (ROI) - S101
Visualize Highly/Loosely Coupled Elements; Microservice Distribution - S102
Advantages
Maintainability, Scalability, Delegation of Teams Responsabilities - S101, S102

Fig. 81.1 Findings from the selected studies

ments and Monitoring, [16]. The authors of S97 used a panoramic view of the grey and informal literature to contextualize scenarios of the migration with a special emphasis on the concern of microservices' granularity. They selected three decision problem scenarios and proposed a solution based on the MAPE-K loop [17] regarding the migration [18].

S98 used the approaches of Domain Driven Design and Bounded Context to support the identification of potential functionalities to be converted into microservices [19]. The authors described the steps they used to the migration, when they also emphasized the use of continuous integration pipeline and continuous delivering [19].

The S99 paper presented the results of a survey with 18 practitioners from which the authors obtained data related to the migration to the microservices architecture. The authors recognized three main phases during the migration: (i) architecture recovery of the legacy system, (ii) architecture transformation from the legacy system to the new architecture, and (iii) the implementation of the new microservice-based architecture system [20].

The authors of study S100 proposed an approach called *Variability-based, Pattern-driven Architecture Migration (V-PAM)*, to support the migration method based on three items; (i) a catalogue of fine-grained service-based cloud architecture migration patterns focusing on multi-cloud scenarios, (ii) a framework to support the pattern selection and composition, and (iii) a variability model to guide the system migration towards a coherent framework [21]. The V-PAM approach considered empirical evidence and data from different migration projects, best practice from cloud architectures and a systematic literature review focusing on the theme [21].

In S101, the authors proposed a framework as a result of the analysis of three different migration processes reported by the interviewed practitioners and respective motivations and challenges faced throughout the migration process [22]. According to the authors, two of the analyzed processes targeted the migration of a legacy monolithic system to a microservice-based through the implementation of the new system from scratch. The third process aims at implementing new features as microservices to replace external services provided by third parties or develop specific features due to new changes that must be implemented to progressively replace the legacy system [22].

The authors of S102 proposed a model-centered process to analyze and visualize the current structure of a legacy software system and dependencies between their components or layers. The goal is to group functionalities into clusters and microservices supported by four different diagrams [4].

81.4.2 Answers to RQ2

The selected studies also provided evidence regarding advantages and challenges related to the migration to microservices (right side of Fig. 81.1).

A non exhaustive list of expected benefits over a traditional monolithic architecture are independence of deployability, language, platform and technology adoption to implement the microservices to accomplish scalability and flexibility from the architectural point of view [23]. The study S101 points out traceability, accountability and auditing as advantages that can be achieved through the use of microservices. It takes into account that the migration accomplished the isolation of business functionalities into microservices that interact among themselves through standardized interfaces [18]. According to S08 [13], the architecture of microservices facilitates the refinement of the limits of business logic, allowing the isolation of units to be tested, making them simpler and easier to understand and maintain.

However, according to S101 many practitioners are not confident to migrate due to the perception that microservice can be a hype and due to the lack of a well-know migration process [22]. In fact, during the migration process, practitioners often deal with common challenges and issues, mainly as a result of their lack of knowledge of best practices and patterns [22]. For example, S101 reported that the main issues associated to the migration are decoupling from the monolithic system, database migration, data splitting, and communication among services.

For medium-sized systems, the adoption of microservices can result in agility, quality improvement, cost reduction, and less time to market. For large cloud systems, they can represent a relevant change in terms of scalability, integration and release frequency [1]. Although microservices can provide substantial benefits, their implementation requires extra machinery and resources, which can impose substantial costs [1]. It is worth to mention that some aspects of this migration are still blurred for practitioners. For example, establishing the appropriate level of granularity and implementing an effective trade-off between size and number of microservices requires in fact expertise to accomplish them [18].

According to [24], the main issue regarding the migration is related to the separation of functionalities intro microservices, in other words, the extraction of microservices from existing monolithic code bases [11], especially in the cases in which the modules are tightly coupled [25]. To accomplish these goals, authors argue that there is the need for tools to automatically to deploy, scale and manage microservices, as well as to log and monitor them. Authors have also mentioned organizational challenges related to the migration as the need for more freedom for teams to implement DevOps tasks [24].

The authors of S96 argued that the migration to a microservice based architecture requires high effort due to the need to analyze every small part of the whole system individually and hence decide what should be converted to microservice [16].

S79 argued that the use of Service Cutter [15], a state-of-the-art tool for microservice decomposition, requires a set of specification artifacts and other specification artifacts containing coupling criteria [15]. In most of the cases, the availability of these documents and respective data is arguable [14].

In S98 [19], the authors argued that the migration to the microservices architecture is not a trivial task. For example, the deployment of the new implemented microservices in the development environment can be difficult, especially for novices. Moreover, the authors explain that service contracts are of vital importance and small changes in the contracts can impact part or the whole system. They suggest the use of service versioning to deal with this problem. The authors conclude the paper stating that microservices is not a silver bullet [19].

The authors of S99 [20] reported that the main challenge was the high level of coupling among the modules and/or components of the legacy system. This implies that the more the modules are coupled among themselves, the more difficult it is to extract functionalities from the legacy system [20]. The identification of service boundaries is another challenge mentioned by the same authors. The need for a *different* mindset for developer was reported by the participants of the study. For example, developers were used to get everything in one single database. In case they needed to get data, they would just query it from the appropriate table. With the distribution of the persistence layer through the microservices, they need to do an HTTP call, including authentication and identification to get this data [20].

In S101 [22], the authors present a list of issues and challenges regarding the migration to microservices based on data obtained from a survey with practitioners. The list is as follows: monolith decoupling, database migration and data splitting, communication among services, effort estimation, DevOps infrastructure requires effort, library conversion effort, peoples minds, expected long-term return on investment (ROI) [22]. As can be seen, many of these items were already mentioned in previous paragraphs, revealing that the results presented in [22] are in line with the other studies.

81.5 Conclusions and Future Work

In this paper, we presented a structured body of knowledge to characterize lessons learned, as well as difficulties and challenges related to the migration from a monolithic software application to a microservice based architecture. The goal of this study is to identify main successful strategies and corresponding challenges reported in the literature during the migration of legacy software systems to microservices.

As future work, we intend to conduct a mapping study and a survey in the industry to identify which guidelines have been adopted by practitioners and compare them with guidelines reported in the literature.

Appendix

Table 81.6 Selected primary studies

Selected studies	Repository
S5 – Extraction of microservices from monolithic software architectures	IEEE Xplorer
S6 – From monolith to microservices: a dataflow-driven approach	IEEE Xplorer
S8 – Methodology to transform a monolithic software into a microservice architecture	IEEE Xplorer
S79 – Microservices identification through interface analysis	Springer
S88 – Service cutter: a systematic approach to service decomposition	Springer
S96 – Highly-available applicationiss on unreliable infrastructure - a microservice architectures in practice	Manual
S97 – Microservices and their design trade-offs a self-adaptive roadmap	Manual
S98 – Migrating to cloud-native architectures using microservices: an experience report	Manual
S99 – Migrating towards microservice architectures: an industrial survey	Manual
S100 – Pattern-based multi-cloud architecture migration	Manual
S101 – Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation	Manual
S102 – Towards the understanding and evolution of monolithic applications as microservices	Manual

References

1. Singleton, A.: The economics of microservices. IEEE Cloud Comput. **3**(5), 16–20 (2016)
2. Kratzke, N., Quint, P.-C.: Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. J. Syst. Softw. **126**, 1–16 (2017)
3. Furda, A., Fidge, C., Zimmermann, O., Kelly, W., Barros, A.: Migrating enterprise legacy source code to microservices: on multi-tenancy, statefulness and data consistency. IEEE Softw. 1–1 (2017) AQ4
4. Escobar, D., Cárdenas, D., Amarillo, R., Castro, E., Garcés, K., Parra, C., Casallas, R.: Towards the understanding and evolution of

monolithic applications as microservices. In: Computing Conference (CLEI), 2016 XLII Latin American, pp. 1–11. IEEE (2016)

5. Reza Bazi, H., Hassanzadeh, A., Moeini, A.: A comprehensive framework for cloud computing migration using meta-synthesis approach. J. Syst. Softw. **128**, 87–105 (2017)
6. Taibi, D., Lenarduzzi, V.: On the definition of microservice bad smells. IEEE Softw. **35**(3), 56–62 (2018)
7. Leymann, F., Breitenbücher, U., Wagner, S., Wettinger, J.: Native cloud applications: why monolithic virtualization is not their foundation. In: Cloud Computing and Services Science, pp. 16–40. Springer, Cham (2016)
8. Fowler, M.: Microservices resource guide. Martinfowler. com. Web 1 (2015)
9. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer Science & Business Media, Berlin/Heidelberg (2012)
10. Petticrew, M., Roberts, H.: Systematic Reviews in the Social Sciences: A Practical Guide. Blackwell Publishing CrossRef Google Scholar, Malden (2006)
11. 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)
12. 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)
13. Acevedo, C.A.J., Y Jorge, J.P.G., Patio, I.R.: Methodology to transform a monolithic software into a microservice architecture. In: 2017 6th International Conference on Software Process Improvement (CIMPS), pp. 1–6 (2017)
14. Baresi, L., Garriga, M., Renzis, A.D.: Microservices identification through interface analysis. In: Service-Oriented and Cloud Computing, pp. 19–33. Springer, Cham (2017)
15. Gysel, M., Klbener, L., Giersche, W., Zimmermann, O.: Service cutter: a systematic approach to service decomposition. In: Service-Oriented and Cloud Computing, pp. 185–200. Springer, Cham (2016)
16. Richter, D., Konrad, M., Utecht, K., Polze, A.: Highly-available applications on unreliable infrastructure: microservice architectures in practice. In: Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on, pp. 130–137 (2017)
17. Betty, H., Rogério, D., Holger, G., Inverardi, P., Magee, J.: Software engineering for self-adaptive systems. Lect. Notes Comput. Sci **5525** (2009) AQ5
18. Hassan, S., Bahsoon, R.: Microservices and their design tradeoffs: a self-adaptive roadmap. In: Services Computing (SCC), 2016 IEEE International Conference on, pp. 813–818. IEEE (2016)
19. Balalaie, A., Heydarnoori, A., Jamshidi, P.: Migrating to cloud-native architectures using microservices: an experience report. In: European Conference on Service-Oriented and Cloud Computing, pp. 201–215 (2015)
20. Paolo Di Francesco, I.M., Lago, P.: Migrating towards microservice architectures: an industrial survey. IEEE Cloud Computing bibtex em 12052018 ainda nao disponivel (2018)
21. Jamshidi, P., Pahl, C., Mendona, N.C.: Pattern-based multi-cloud architecture migration. Softw. Pract. Exp. **47**(9), 1159–1184 (2017)
22. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation. IEEE Cloud Comput. **4**(5), 22–32 (2017)
23. Lewis, J., Fowler, M.: Microservices: a definition of this new architectural term (2014). http://martinfowler.com/articles/microservices. html (cit. on p. 26) (2017)
24. Kalske, M., Mkitalo, N., Mikkonen, T.: Challenges when moving from monolith to microservice architecture. In: Current Trends in Web Engineering, pp. 32–47. Springer, Cham (2017)
25. DAgostino, D., Danovaro, E., Clematis, A., Roverelli, L., Zereik, G., Galizia, A.: From lesson learned to the refactoring of the DRIHM science gateway for hydro-meteorological research. J. Grid Comput. **14**(4), 575–588 (2016)

AUTHOR QUERIES

AQ1. Please provide organization details for author “H. C. da Silva Filho”.
AQ2. Chapter title is mismatch between chapter opening page and table of contents, so we have retained as per chapter opening page. Please check if okay.
AQ3. Please check if inserted citation for “Tables 81.3, 81.4, and 81.5” are okay.
AQ4. Please provide volume number for Ref. [3].
AQ5. Please provide page range for Ref. [17].