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)

Analysis of the Criteria Adopted in Industry to Extract Microservices

Luiz Carvalho*, Alessandro Garcia*, Wesley K. G. Assunção†, Rafael de Mello*, Maria Julia de Lima‡
*Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Rio de Janeiro, Brazil
{lmcarvalho, afgarcia, rmaiani}@inf.puc-rio.br
†Federal University of Technology - Paraná (UTFPR). Toledo, Brazil
wesleyk@utfpr.edu.br
‡ Tecgraf Institute, Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Rio de Janeiro, Brazil
mjulia@tecgraf.puc-rio.br

***Abstract*—A microservice architecture is expected to provide a better modularization and management of small and autonomous services. Other expected benefits include increased availability and time to market. There is a growing interest of both industry and academia on streamlining the migration of existing systems to a microservice architecture. However, the success of this migration is largely dependent on the use of appropriate criteria for extracting microservices from a code base. Recent studies indicate the selection and decomposition of microservices represent the main challenge along the migration. Academic techniques tend to support the extraction of microservices with either one or two conventional criteria, namely coupling and cohesion. There is limited knowledge on the criteria actually considered as useful by practitioners. Thus, we have performed an exploratory online survey with 15 specialists experienced on migrating systems to a microservices architecture. In particular, we question the relative usefulness of seven possible criteria for supporting decision-making along microservice extraction. The participants were also questioned about tools they have used, their limitations, and whether the decisions on extracted microservices were considered unsuccessful. Overall, the survey results suggest academic techniques do not totally satisfy the needs of practitioners. Practitioners often need to consider simultaneously at least four dominant criteria as well as their trade-offs to support their decisions. Most practitioners consider existing tooling support insufficient or even irrelevant to support their microservice extraction decisions.**

***Index Terms*—microservices, reengineering, extraction, survey, industry**

I. Introduction

Microservices have been successfully adopted for developing software systems in successful companies, such as Netflix [1] and Uber [2]. Microservices are small and autonomous services that work together [3]. A microservice is expected to have fine granularity [4]. It is also expected to be autonomous: (i) it should consist of a service highly independent from others, and (ii) it should enable an independent choice of technologies, such as the programming language, the database, the communication protocol, and the like. A number of benefits are typically associated with a microservice-based architecture, such as reduced maintenance effort, increased availability, improved innovation, continuous delivery, DevOps enabler, scalability, and reduced time to market [5], [6].

Not rarely, microservices are not developed from scratch, but they result from the migration of existing systems [1], [2], [5], [7], [8]. Given the claimed benefits of microservices, there is a growing interest of both industry and academia on streamlining the migration to a microservice architecture. However, the migration from a monolithic architecture to microservices is perceived as challenging by developers who have experienced it [3], [5], [6]. In particular, the success of this migration is largely dependent on the use of appropriate criteria for extracting microservices from a code base.

In this work, microservice extraction comprehends the task of deciding whether and which parts of existing system functionality(ies) will be migrated to a microservice, by the selection, decomposition, and reuse of existing system parts. Academic approaches for supporting the microservice extraction have been proposed [3], [9]–[11]. Most often, they tend to support the extraction of microservices with either one or two conventional criteria, namely coupling and cohesion [9], [10]. A few others tend to solely use the information on database schema [3], [11] to support extraction decisions.

There is limited knowledge on whether academic techniques (e.g., [3], [9]–[11]) are aligned with how practitioners perform microservice extraction. Existing studies in the industry often focus on reporting benefits and challenges on the migration to a microservice architecture (e.g., [1], [2], [5]–[8], [12]). For instance, they point the selection and decomposition of microservices are often the most challenging activities [5], [12]. However, it remains unclear what are the extraction criteria perceived as useful by practitioners along these tasks. Differently from many software engineering techniques, microservices emerged in the industry and not in the academia[1]. This fact possibly slowed the process for researchers to fully understand the peculiarities of this architecture style.

To address this gap, this paper presents an investigation of the criteria perceived as useful by practitioners for extracting microservices. To achieve this aim, we perform an exploratory study based on an online survey. The basis for our survey is a set of seven criteria mentioned in articles written by either researchers or practitioners. We want to understand the degree of

[1]https://martinfowler.com/articles/microservices.html

2575-4793/19/$31.00 ©2019 IEEE
DOI 10.1109/CESSER-IP.2019.00012

usefulness (if any) of these criteria. We also questioned which tools are actually used by practitioners while analyzing the criteria for microservice extraction. The survey was distributed to industry specialists, which included developers, architects, project managers, and industry researchers. Our survey relies on the participation of 15 specialists from different countries located in North America, South America and Europe. They all have considerable practical experience on migrating existing systems to microservices architecture. These participants also have extensive experience in software development.

Overall, the survey results suggest academic solutions do not satisfy the industrial needs. For instance, practitioners often consider simultaneously at least four dominant criteria as well as their trade-offs to support their decisions. Academic solutions tend to only support the analysis a few criteria, which makes it hard for developers to reveal and analyze trade-offs with unsupported criteria. This is one of the reasons why practitioners reported they consider existing tooling support insufficient or even irrelevant for enabling well-informed decisions on microservice extraction. In summary, our survey provides a step further on a more realistic understanding of the relative usefulness of microservice extraction criteria in practice, from the point of view of industry specialists. Thus, we expect our survey results encourage researchers and practitioners to work more closely to design appropriate techniques and tooling support for microservice extraction.

The remainder of this paper is organized as follows. We present an illustrative example in Section II, the set of investigated criteria in Section III, our survey design in Section IV, results and discussions in Section V, threats to validity in Section VI, limitations of related work in Section VII, and conclusions in Section VIII.

II. Microservice Extraction: An Illustrative Example

In spite of the potential benefits, the decomposition of an existing system into microservices is complex. We present a real case to illustrate how a particular criterion can affect the (mis)decision of extracting microservices from an existing system. This example comes from a system (from herein called "monolithic system") with limited modularization and no documentation. Maintainers of this system reported its notable limitations to incorporate new technologies. Some of the characteristics (and other ones) led to the partial migration of a monolithic structure to a microservice architecture.

The monolithic system was initially developed assuming a single process execution with communication made by local function calls. For the partial migration to a microservice architecture, engineers and developers decided to identify and extract some functionalities to microservices, thereby introducing network communications among them. Even though the extracted microservices fostered the decoupling of functionalities, a drawback was observed afterwards in the response time of some requests to the REST API. The analysis of the code extracted from the original monolithic system revealed some inner loops producing remote communication overhead. Developers decided to add a cache layer in the calls of this API, reducing roundtrips to dependencies and achieving the expected time response by the users.

This example illustrates how certain criteria affect the outcome of the extraction process. The decoupling among system functionalities was the dominant (maybe single) criterion considered along extraction. However, the result decomposition indirectly affected communication overhead, deteriorating system performance. The lack of reasoning upfront about other important criteria (communication overhead, in this case) and their trade-offs led to the microservices' unsuccess. A better understanding of these criteria is critical to support decision-making process during the microservice extractions.

III. On the Criteria for Microservice Extraction

To define a representative set of criteria, we searched in the technical literature the criteria explicitly reported for migrating existing systems to a microservice architecture. This review involved reading and analyzing papers cited by two mapping studies on microservices [13], [14]. As a result, we compiled an initial set of seven criteria, including (i) the ones used in academic techniques, and (ii) the ones mentioned in technical literature, albeit not necessarily supported by existing academic automated techniques. Our goal was to understand their usefulness from the perspective of practitioners with experience on microservices migration processes. These criteria, included in our survey with specialists (Section IV), are described below.

Coupling. Coupling comprehends the manner and degree of interdependence between software modules or functionalities [15]. Newman mentions that microservices should be decoupled from each other as much as possible [3]. Different coupling metrics are used by existing techniques (e.g., [9], [10]) for microservice extraction. Coupling is the most cited criterion in the literature [13], [14]. In addition, case studies select coupling as the dominant (sometimes, single) criterion along the microservices migration process (e.g., [16]).

Cohesion. Cohesion is the manner and degree in which inner elements – data and behaviors – of a single module (or functionality) are related to each other [15]. A microservice is expected to modularize a functionality (e.g., a domain entity) with highly cohesive inner elements. Therefore, cohesion is an important aspect for microservice architectures. Existing academic techniques indeed consider cohesion to recommend the extraction of microservices from existing systems [10]. We also found case studies of microservice-driven migrations reporting cohesion as a relevant criterion [16]. However, cohesion is used as a secondary criterion in some academic solutions; coupling is considered the main criterion.

Overhead in the communication of extracted microservices. Overhead is related to the amount of time a system would spend on performing actions not directly addressing the user needs [15]. Communication overhead concerns the negative impact of extracting microservices on time spent along future microservice communication, which was originally performed locally (e.g., via function calls) in the "monolithic

system". The derived microservices will need to keep communicating by protocols such as HTTP and AMQP, therefore, this communication may result in some penalties, possibly prohibitive ones, to the system performance [3], [17], [18]. That is the reason why communication overhead might need to be considered beforehand, through microservice selection activities. It should be noted that coupling and overhead in the communication of extracted microservices are different criteria. Coupling is the degree between different elements while overhead is time consumed in network communication.

Potential of reuse. Reuse is the use of already developed assets in the solution of different problems [15]. For example, microservices, software product lines, frameworks, and libraries aim to support different levels of reuse. Each extracted microservice could be reused by two or more of its callers. The cost of microservice extraction can be justified by actual reuse in the short or long term. Higher the potential of microservice reuse, the better. That is why practitioners may consider this criterion when selecting and decomposing microservices. There is another way of considering reuse along the migration process: only built microservices that can reuse parts of the existing code base. Previous studies carried out migrations to microservices in order to maximize code reuse [5], [7].

Data dependencies in the database schema. Data entities have dependencies in a database schema [19]. Moreover, the database structure could be considered the greatest source of dependencies for a wide range of systems. Migrating to the microservices architecture may lead to splitting the database into smaller databases. Database decomposition can achieve data coupling reduction and increase microservices' independence. Such an independence make it possible the selection of different database technology for each microservice. In addition, within the database schema it is possible to find valuable information about the domain entities (e.g, their relations, constraints, and the like). The analysis of the database schema is mentioned as a way of possibly starting the microservice extraction process [3], [11].

Impact on software requirements. Requirement is the software capability needed by a user to solve a problem or achieve an objective [15]. In microservices migration process, practitioners may also consider the possible impact of candidate microservices on certain non-functional (or functional) requirements. For example, non-functional requirements as security may influence which parts of existing systems could be extracted. Functional requirements may provide a better understanding of the domain concepts.

High-level criteria from visual models. Some decomposition criteria can be more easily analyzed from visual models. The migration of an existing system to a microservice architecture can be seen as a combination of reverse engineering and re-engineering. Reverse engineering [20] may use graphical resources to build representations of the existing systems in higher levels of abstraction. Thus, visual representations, such as UML, could be useful to derive architectural on detailed designs of the system. These representations may be used to support analyses of high-level criteria that impact on the selection and/or decomposition of microservices. For instance, two functionalities with a strong architectural dependency may be a criterion used to give up on their modularization as two microservices. Moreover, behavioral models of the system may be used to decide on the possible boundaries of a candidate microservice. Our literature review revealed a previous study that consider use cases to define some criteria for microservices extraction [18].

IV. Survey Design

This section presents the design of our survey based on the potentially useful criteria used along microservice extraction.

A. Goal, Population and Sample

We conducted an exploratory survey [21] with the goal of understanding what criteria are considered useful. Specialists on microservice migration participated in the survey. To achieve ours goal, we asked the subjects to evaluate the perceived relevance of the criteria presented in Section III. Moreover, we inquired the subjects which techniques and tools they use when applying each criterion.

The target of our survey is composed of specialists with a background in migrating existing systems to microservices. We developed a search plan [22], aiming at identifying a representative sample of this target. We selected a source of sampling composed of two mapping studies on microservices [13], [14]. Then, we searched in this source by works addressing the migration of existing systems to microservice architecture. In the following step, we performed a snowballing search [23] in the pieces of work that cited the referred mapping studies. For performing this search, we used Google Scholar[2].

The survey was sent to authors of scientific papers. Some of these authors actively participated in the migration process to microservices architecture. They have played key roles, such as developers or architectures. In this way, they represented the target to our survey. Moreover, we are also invited subjects (i.e., participants) of the empirical studies (reported in the analyzed papers) to take part of the survey when the paper authors agreed is distributing the survey to the corresponding study's subjects. After execution, our search plan resulted in the recruitment of 70 subjects. The survey was executed during January 2019. From the 70 subjects recruited, 15 participants responded, resulting in a participation rate of 21.23%.

B. Instrumentation

In this study, we applied an online questionnaire for gathering subjects' data. The questionnaire items are divided into three groups. The first group is composed of questions for characterizing the survey participants. This group of questions is presented in Table I. Among others, we asked the subjects' academic background, development experience, and position in the current job. More specifically, we asked about their background in migrating existing systems to microservices.

The second group of questions (Table II) aims at gathering the perceived utility of criteria presented in Section III. We

[2]scholar.google.com

TABLE I
SURVEY QUESTIONS ABOUT CHARACTERIZATION OF RESPONDENTS

Question	Type
Do you want to receive future information about the survey results?	Choice: Yes, No
Name	Open Question
Email	Open Question
What is your academic background?	Choice: HS, Grad, Master, PhD, Other
How long have you been developing software? (years)	Positive Number
What is your position in your current job?	Open Question
How many migration processes to a microservice architecture did you only participate in the past (e.g., in architecture decisions, programming tasks, and others)?	Positive Number
How many migration processes to a microservice architecture are you currently participating (e.g., in architecture decisions, programming tasks, and others)?	Positive Number
How many migration processes to a microservice architecture did you only observe in the past (e.g., examination, analysis, review, consulting, and others)?	Positive Number
How many migration processes to a microservice architecture are you currently observing (e.g., examination, analysis, review, consulting, and others)?	Positive Number
What is (are) the domain(s) of the systems that underwent migration(s) to a microservice architecture with your involvement?	Open Question

TABLE II
SPECIFIC QUESTIONS TO ALL THE CRITERIA OF SECOND GROUP

Criterion	Question	Type	Answer
Coupling	How did you measure coupling?	Checkbox	1. The structure of the source code 2. Program execution 3. Others
Cohesion	How did you measure cohesion?	Checkbox	1. The structure of the source code 2. Program execution 3. Others
Communication Overhead	How did you measure overhead in future network communication between extracted microservices?	Checkbox	1. The structure of the source code 2. Program execution 3. Others
Reuse Potential	How did you measure reuse?	Checkbox	1. The syntax of the code 2. Code duplication 3. Others
Requirements Impact	What are the requirements considered (if any)?	Open, Likert Scale	Textual
Visual Models	What was (were) visual representation(s) used?	Open	Textual

used a five-point Likert scale associated with the following levels of usefulness, from the lowest to the highest: 1) Not useful - it was not useful at all, 2) Slightly useful - it was only useful to a small degree, 3) Moderately useful - it was not considerably usefulness, 4) Useful - it was considerably usefulness, 5) Very useful - it was indispensable.

To avoid misinterpretations, we provided in the questionnaire a formal definition of each criterion evaluated (see Section III). We also asked the participants to justify their answers. We also want to understand what is used to analyze or measure each criterion. For this purpose, we asked the tools and techniques they use in each criterion evaluated. We also asked whether developers consider these tools sufficient to support the migration process, as shown in Table III.

TABLE III
COMMON QUESTIONS TO ALL THE CRITERIA OF SECOND GROUP

Question	Type
Feel free to justify the usefulness (or not) of the criteria in the decision-making process	Open Question
What tool(s) was (were) used for measuring or analyzing the criteria?	Open Question

For each criterion investigated, we also applied particular questions, based on its characteristics. Coupling, cohesion and overhead are usually measured through static and dynamic analysis [24]. In this way, we asked whether subjects use one of these approaches or both. For reuse, we asked whether this criterion is measured through the syntax of the source code and by the incidence of duplicated code. Regarding software requirements, we questioned whether non-functional and functional requirements are used. Regarding visual models, we inquired about the type of artefacts commonly used to represent the system in higher levels of abstraction.

The third group of questions are about the sufficiency of the tools to support the extraction of microservices. We also asked about cases where the extraction may have failed. These questions are all open answer and are presented in Table IV.

TABLE IV
QUESTIONS IN THE THIRD GROUP

Question
Do you believe that existing tools are sufficient to support the migration process?
Was there any case that the extraction of the microservice was not successful (led to a high overhead communication between microservices, data inconsistency, or some other problems)?

V. RESULTS AND DISCUSSIONS

Next we present the results of the application of our survey to a group of 15 specialists. The experience of the respondents can be evidenced by the fact they all have both previous and ongoing experience with microservices migration processes. For instance, the vast majority of the respondents had already concluded their participation in at least two migration processes (a mean of 2.7 processes). Moreover, most of them are

TABLE V
ON THE CRITERIA USEFULNESS: PARTICIPANT RESPONSES

Criterion	Responses															Median
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	
Coupling	3	3	3	4	4	4	4	5	4	4	5	5	4	5	4	4
Cohesion	5	4	5	4	4	5	4	5	2	4	5	5	5	3	3	4
Communication Overhead	2	2	1	4	3	4	2	3	4	3	5	1	4	3	2	3
Reuse Potential	4	4	4	3	5	2	4	5	1	2	1	4	2	2	5	4
Database Schema	4	4	1	3	2	4	5	3	2	1	5	5	1	4	3	3
Requirements Impact	4	4	4	3	3	5	5	3	4	1	4	5	5	4	4	4
Visual Models	3	2	5	3	2	2	4	2	1	5	3	5	3	5	3	3

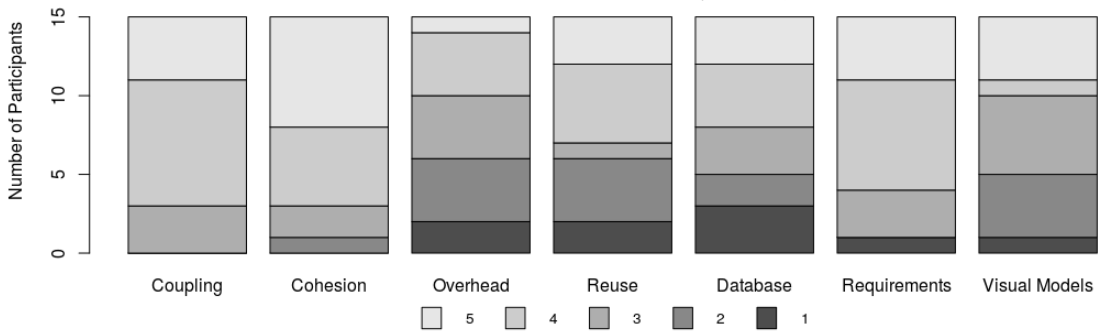

Fig. 1. Criterion Usefulness: Distribution of Responses

actively participating in at least one project (a mean of 1.6) where the microservices extraction is currently underway.

The respondents have extensive experience in software development. The time in years the participants have been developing software is considerably high: (i) the mean is 17.4 years and median is 18 years, and (ii) with a maximum of 35 years and a minimum of 3 years. Our sample of respondents is also well diversified in terms of the roles they play in their projects: developers (33.3%), architects or engineers (20%), team leaders (20%), and industry researchers (26.7%).

Table V and Fig. 1 presents the answers of the participants with respect to the usefulness of each extraction criterion. As explained in Section IV.B, we used a five-point Likert scale to enable the respondents to classify each criterion from not useful at all (1) to very useful (5). The analysis of the medians (last column) of each criterion (rows) in Table V reveals which criteria are useful (median equal to 4) and moderately useful (median equal to 3). The columns show the answers of each participant with their IDs varying from #1 to #15. Fig. 1 provides a different perspective on the responses by showing which criteria concentrates more positive, neutral, or negative scores. In the following, we discuss first the usefulness of each criterion. We start discussing those criteria considered as the most useful by the practitioners. After that, we explore other dimensions in the obtained responses.

Cohesion as the most useful criterion, but proper tool support is missing. Cohesion was considered the most useful criterion: it has the highest concentration of "very useful" answers, represented by the light grey color in Fig. 1. This criterion presented the median equal to 4 (Table V). This was a surprising result as we expected that coupling would outperform cohesion. Coupling is usually prioritized in the conventional decomposition of software modules, while cohesion tends to be used to confirm the incidence of modularity flaws. One participant justified that *"A highly cohesive functionality should make a natural unit that is very easy to spot and derive the microservice boundaries"*. Cohesion was described by the majority of the participants as more useful than coupling to support decisions on microservice extraction. Multiple respondents mentioned cohesion is the key criterion to identify domain entities by domain driven design, a typical method used in industry to identify microservice candidates.

Despite the relevance of cohesion, surprisingly, only 33.3% respondents reported they use some tool, such as IntelliJ[3], to apply this criterion. These results suggest that cohesion is predominantly analyzed manually. One of the reasons it that some practitioners mentioned they need to understand the cohesion from a dynamic analysis perspective. Actually, more than 30% of the participants reported using the system execution to analyze cohesion. One of the respondents even mentioned he tries to infer cohesion while debugging the code with IntelliJ. Some participants use static cohesion measurement, but most of them do it by convenience as most of the existing tools support only static cohesion measurement.

Coupling is also an useful criterion, but again dynamic analysis support is missing. Coupling also achieved a median of 4 (Table V). It was the second criterion in terms of concentrating "very useful" and "useful" answers (Fig. 1).

[3]https://www.jetbrains.com/idea/

Few participants (26.6%) again reported using some tool to support coupling analysis, such as IntelliJ and SonarQube[4]. Even though many participants reported to use the program structure for coupling analysis, almost 40% of the respondents mentioned other strategies. In particular, 33.3% mentioned they need to infer actual coupling of functionalities from a dynamic analysis perspective while considering a functionality to be extracted to a microservice. They need to understand which of the static dependencies can have indeed an impact on possible performance bottlenecks if a functionality is modularized as a microservice. However, the use of dynamic coupling analysis is always challenging as it can generate a large output, and clearly requires proper tool support [25]. One participant also mentioned he usually analyzes the change history of functionalities to reveal their co-changes that imply logical coupling (not capture by static dependencies).

Impact on requirements: functional and non-functional requirements are equally important. Impact on requirements was the third most relevant criterion. The distribution of answers for this criterion in the Likert scale was quite similar to the one observed for coupling. Most of the participants perceived this criterion as "useful" or "very useful", but there was one participant that gave the lowest score for this criterion. He mentioned the requirements specification of the existing system did not predict at all relevant requirements (either functional or non-functional) for the microservices being considered in the migration process.

We expected that non-functional requirements would appear much more frequently than functional requirements. Functional requirements were by far the most quoted type of requirement considered along the microservice extraction: 77% of the valid responses considered them as "very useful". Functional requirements are used to infer the domain concepts which can be isolated in microservices. Performance was perceived as the most useful (median of 4) non-functional requirement. Respondents were clearly concerned with performance deterioration. A total of 86% of participants rely on explicit description of requirements in the migration process.

Reuse opportunities are considered on microservice extraction, albeit not unanimously. Reuse also achieved a median of 4 in the responses. One participant mentioned the practice of reuse *"is indeed a key driving factor for the migration to microservices architecture; it is important to promote the reuse of the extracted microservice by other systems"*. However, the relevance of reuse was not unanimous. There was a distribution of different answers across the five points in the Likert scale. While 8 participants gave the highest scores (either 5 or 4), 6 partipants gave the lowest scores (either 1 or 2) and 1 participant gave a borderline score (3).

Some participants argue there are other more effective ways to achieve reuse. One participant quoted that: *"Though arguably a good reason to create new microservices, I haven't witnessed any project in which its main motivation was reuse. The most common solution for achieving reuse was the use of software libraries instead."*. Others argue that the existing monolithic code was so intermingled that opportunity for reuse could not even be considered. Among the participants, 40% reported they use tools, such as IDEs and its plugins, for supporting reuse analysis. Code duplication (26.6%), code syntax (46.6%), and other factors were adopted in the decision-making process to find reuse opportunities while deciding on extracting microservices. However, 60% of the respondents stated that reuse analysis is predominately manual.

Communication overhead is a secondary criterion as participants often postpone its analysis. Microservices may need to communicate with other microservices or subsystems. Then, the overhead on microservice communication was expected to be a primary criterion while deciding for microservice extraction. However, it was overall perceived as moderately useful with a median equal to 3. Fig. 1 shows it has the highest distribution of answers in the Likert scale. Some comments state that the other benefits of extracting microservices outweight the possible drawback of increased communication overhead.

Only five of the respondents considered communication overhead is "very useful" or "useful". These respondents were the same who classified performance as an important non-functional requirement. We observed that many participants postpone the analysis of communication overhead, *i.e.*, they tend to postpone it for after the extraction of the microservice(s). However, some participants mentioned they try to predict if the communication overhead will be prohibitive, and/or which kind of microservice decomposition will minimize the overhead. Some participants reported the use of different tools to support overhead analysis, such as Intellij IDEA, JMeter, and debugging tools. Runtime analysis via logs and microservices monitoring tools were also mentioned.

Surprisingly, dependencies on database schemas is frequently a neglected criterion. Opinions on the usefulness of this criterion also varied as much as communication overhead. This was a surprising result. Proponents of microservices often state the each microservice should have its own tiny database. Therefore, we expected that data dependencies in database schema would be often used in the extraction of microservices. However, our respondents seem to decompose microservices using different strategies with respect to the database.

Half of the respondents seem to follow recommendations of proponents of microservices architectures. In this case, most of them stated they use tools associated with relational database management systems to reason about the database schema decomposition. These participants also mentioned database schema is useful for identifying independent domain concepts.

High-level criteria from visual representations used for determining microservice boundaries. Only five (one third) of the respondents considered high-level criteria, derived from visual representations, as "very useful" or "useful". One third considered their usefulness as moderate, while the other one third qualified them as "not useful" or "slightly useful". The most common representations are class diagram, dataflow models, and use cases. Regardless the type of representation,

[4]https://www.sonarqube.org/

the most common use of it was to support the definition and analysis of criteria to determine the boundaries of the microservices being considered for extraction.

All practitioners often use at least four dominant criteria simultaneously. Most of the academic solutions investigated in our literature review (Section III) suggest some dominant criteria for extracting microservices. Most of them: (i) support the analysis of one or two criteria only [3], [9]–[11], and (ii) suggest to use coupling and cohesion as the dominant criteria. In fact, the vast majority of the proposed techniques for supporting microservice extracting rely on coupling or on a combination of coupling and cohesion only [9], [10]. Moreover, they only support the analysis of coupling and cohesion of functionalities already structured as separated modules. However, our survey revealed that existing systems are often legacy systems. Thus, some functionalities being considered to be extracted as microservices tend to be scattered and tangled with each other in the implementations of existing modules. They are not modularized in the existing system, which hampers the use of academic techniques for supporting microservice extraction.

To make the matters worse: most of the respondents (9 out of 15) reported at least four criteria were simultaneously used to support decisions on the extraction of each microservice. This result can be observed in Table V: these respondents are the ones who pointed out at least four criteria "very useful" or "useful". Table V shows that four of these respondents qualified more than four criteria "very useful" or "useful". The 6 remaining respondents used 3 dominant criteria. However, the results of our suggest that academic solutions are currently oversimplifying the problem of microservice extraction. Practitioners need to consider to maximize the satisfaction of these multiple criteria and deal with various emerging trade-offs.

Most specialists believe existing tools are not sufficient. We questioned about whether existing tools are sufficient to support the decision-making process of extracting microservices. A tally of 53,3% of the participants answered no. Only 26,7% said that tools are sufficient, while 20% stated that do not believe in tools to support the migration process. This was the quote of one of the participants: *"The tools themselves cannot provide a path clear enough to allow easy decisions towards extraction or not."* In the opposite direction, other participant said: *"Tools could help to provide more confidences on the effectiveness of these decisions"*. We also inquired whether any extraction of certain microservices was unsuccessful due to some criterion neglected in the process. Surprisingly, 40% of the respondents answered yes. They stated that the lack of synthesized information about the relevant criteria was one of the reasons to overlook a certain criterion in spite of their ultimate importance.

VI. Threats to Validity

The survey questionnaire is composed of a large number of questions, which may discourage the subjects' participation. In this sense, before running the survey, we first invited other researchers to review the questionnaire. Based on their feedback, we then conducted a pilot with real subjects. In this pilot, we observed an acceptable participation rate for the context of surveys in the field.

The small sample of respondents in the survey is a threat to validity. However, most of survey participants declared significant experience on migrating systems to microservice. This achievement was possible due to the fact we followed a formal recruitment strategy for identifying highly qualified subjects to participate in the survey (see Section IV). This sampling process resulted in the recruitment of 70 individuals involved with the research and practice of implementing microservices. A participation rate of 21.5% is quite high for online surveys of this kind, which usually ranges from 3% to 10%.

VII. Related Work

In the migrating of existing system to a microservice architecture, some techniques extract source code information and properties to recommend microservices. Mazlami et al [9] present a strategy to decompose systems into microservices considering three coupling metrics. These metrics are used to weight a graph where nodes represent system classes. Edges are associated with weights provided by the coupling metrics. The components provided by a clustering algorithm are used to recommend microservices to the developer.

Other previous study considers the production and use of diagrams to understand the legacy systems for suggesting microservices. The idea is to separate and group Enterprise Java Beans (EJB) according to the type of data it handles [11]. In other words, this is a strategy centered on a model-based criterion. Moreover, Newman [3] presents recommendations of microservice extractions based on certain criteria associated with the database schema. Jin et al [10] propose a functionality-oriented microservice extraction method by monitoring system execution traces and clustering them. In spite of promoting the use of dynamic execution of the system, it is still limited because it only considered coupling and cohesion. As observed in our survey, practitioners need to explore various combinations of criteria in order to make successful microservice extractions in industry cases. Unfortunately, these combinations are not either supported or used in existing techniques and methodologies found in the literature review.

Francesco et al. [26] interviewed and applied a questionnaire to developers. Their goal was to understand the performed activities, and the challenges faced during the migration. They reported what are the existing system artifacts (e.g., source code and documents) the respondents used to support the migration. The main reported challenges were: (i) the high level of coupling, (ii) the difficulty of identifying the service boundaries, and (iii) the microservices decomposition. However, they did not specifically analyzed the usefulness of the extraction criteria addressed in our survey.

Taibi et al. [6] also conducted a survey with the objective of elucidating motivations that led to the microservices migration process and what were the expected returns. The main motivations were the improvement of maintainability, scalability, and delegation of team responsibilities. In addition,

difficulties were cited in this process, such as decoupling from the monolithic system, followed by migration, and splitting of data in legacy databases.

VIII. Conclusions

In this paper, we reported a survey performed with specialists to assess what criteria are useful to extract microservices during the migration to a microservice architecture. We questioned how the participants measure and analyze (with or without tools) each criterion. We also inquired them if: (i) existing tooling support is sufficient, and (ii) unsuccessful microservice extractions occurred. To identify potential respondents, we searched for studies in two mapping studies about microservices migration. We also made a snowballing search starting from the mappings.

Even though there were some variations across participants' answers, the results revealed that participants consider criteria related to modularity – *i.e.*, coupling, cohesion, and reuse – and requirements impact as relevant. The other three criteria are commonly seen as moderate, albeit considered "useful" or "very useful" by certain respondents. In other words, there is no criterion that can be claimed to be not useful. Their degree of usefulness clearly varies from a context to another. Moreover, effective decisions on microservices extraction are clearly far from being simplistic. Our results suggest four criteria are usually considered simultaneously to support decisions. This finding questions the practicality of academic solutions, which generally consider only one or two criteria.

In fact, existing techniques and tools were seen as insufficient. Mistaken decisions on microservice extractions are mentioned to be often related to the lack of synthesized information about the relevant criteria and their trade-offs. All these findings of our survey suggest researchers and practitioners to work more closely. Otherwise, it is unlikely we will able to design appropriate techniques and tooling support for microservice extractions. There are many unaddressed questions: how to automatically identify possible trade-offs among the seven criteria based on the legacy code? how to better combine static and dynamic analysis to reveal such trade-offs? how to anticipate information on potential communication overheads before the microservice extractions or in a step-wise manner?

As future work, we plan to perform interviews and additional analysis, such as the identification of existing patterns, in the answers. For example: how the degree of developers' experience related to their criteria prioritization? what are other new criteria (beyond the seven ones) spontaneously mentioned in their responses? Moreover, so far we have 15 answers. However, the survey is still open. We are expecting to receive more answers. These additional answers may help us to further confirm or reveal new insights.

Acknowledgment

This research was funded by CNPq grants 434969/2018-4, 312149/2016-6, and 408356/2018-9, CAPES grant 175956, FAPERJ grant 22520-7/2016.

References

[1] C. Watson, S. Emmons, and B. Gregg. (2015) A microscope on microservices. [Online]. Available: http://techblog.netflix.com/2015/02/a-microscope-on-microservices.html
[2] S. Fowler, *Production-Ready Microservices*, 2016.
[3] S. Newman, *Building Microservices*, 1st ed. O'Reilly Media, 2015.
[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, *Microservices: Yesterday, Today, and Tomorrow*. Cham: Springer International Publishing, 2017, pp. 195–216.
[5] W. Luz, E. Agilar, M. C. de Oliveira, C. E. R. de Melo, G. Pinto, and R. Bonifácio, "An experience report on the adoption of microservices in three brazilian government institutions," in *SBES*. New York, NY, USA: ACM, 2018, pp. 32–41.
[6] D. Taibi, V. Lenarduzzi, and C. Pahl, "Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation," *IEEE Cloud Computing*, vol. 4, no. 5, pp. 22–32, 2017.
[7] W. Scarborough, C. Arnold, and M. Dahan, "Case study: Microservice evolution and software lifecycle of the xsede user portal api," in *Conference on Diversity, Big Data, and Science at Scale*. New York, NY, USA: ACM, 2016, pp. 47:1–47:5.
[8] J. Gouigoux and D. Tamzalit, "From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture," in *ICSAW*, 2017, pp. 62–65.
[9] G. Mazlami, J. Cito, and P. Leitner, "Extraction of microservices from monolithic software architectures," in *ICWS*, 2017, pp. 524–531.
[10] W. Jin, T. Liu, Q. Zheng, D. Cui, and Y. Cai, "Functionality-oriented microservice extraction based on execution trace clustering," in *ICWS*, 2018, pp. 211–218.
[11] D. Escobar, D. Crdenas, R. Amarillo, E. Castro, K. Garcs, C. Parra, and R. Casallas, "Towards the understanding and evolution of monolithic applications as microservices," in *CLEI*, 2016, pp. 1–11.
[12] L. P. Tizzei, M. Nery, V. C. V. B. Segura, and R. F. G. Cerqueira, "Using microservices and software product line engineering to support reuse of evolving multi-tenant saas," in *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*, ser. SPLC '17. New York, NY, USA: ACM, 2017, pp. 205–214.
[13] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study," in *CLOSER*, 2016, pp. 137–146.
[14] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *SOCA*, 2016, pp. 44–51.
[15] *ISO/IEC/IEEE 24765: 2017(E): ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary*. IEEE, 2017.
[16] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From monolithic to microservices: An experience report from the banking domain," *IEEE Software*, vol. 35, no. 3, pp. 50–55, 2018.
[17] H. Knoche, "Sustaining runtime performance while incrementally modernizing transactional monolithic software towards microservices," ser. ICPE '16. New York, NY, USA: ACM, 2016, pp. 121–124.
[18] Namiot and M. Sneps-Sneppe, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.
[19] R. Elmasri and S. Navathe, *Fundamentals of Database Systems*, 6th ed. USA: Addison-Wesley Publishing Company, 2010.
[20] E. J. Chikofsky and J. H. Cross, "Reverse engineering and design recovery: a taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13–17, 1990.
[21] J. Linaker, S. M. Sulaman, R. Maiani de Mello, M. Höst, and P. Runeson, "Guidelines for conducting surveys in software engineering," 2015.
[22] R. M. de Mello and G. H. Travassos, "Surveys in software engineering: Identifying representative samples," in *ESEM*. ACM, 2016, p. 55.
[23] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *EASE*. New York, NY, USA: ACM, 2014, pp. 38:1–38:10.
[24] A. Tahir and S. G. MacDonell, "A systematic mapping study on dynamic metrics and software quality," in *ICSM*, 2012, pp. 326–335.
[25] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A systematic survey of program comprehension through dynamic analysis," *IEEE Transactions on Software Engineering*, vol. 35, no. 5, pp. 684–702, Sep. 2009.
[26] P. D. Francesco, P. Lago, and I. Malavolta, "Migrating towards microservice architectures: An industrial survey," in *ICSA*, 2018, pp. 29–2909.