The Journal of Systems & Software 206 (2023) 111826

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

CharM — Evaluating a model for characterizing service-based architectures☆

Thatiane de Oliveira Rosa [a,d,*], Eduardo Martins Guerra [b], Filipe Figueiredo Correia [c], Alfredo Goldman [a]

[a] Mathematics and Statistics Institute of the University of São Paulo - IME-USP, Matão Street, 1010, Butantã, São Paulo, SP, 05508-090, Brazil
[b] Computer Science Faculty of the Free University of Bozen-Bolzano, Bolzano, Alto Adige, Italy
[c] Faculty of Engineering of the University of Porto, INESCTEC, Porto, Portugal
[d] Federal Institute of Tocantins, Paraíso do Tocantins, TO, Brazil

ARTICLE INFO

Article history:
Received 15 July 2022
Received in revised form 15 May 2023
Accepted 22 August 2023
Available online 26 August 2023

Keywords:
Software architecture
Service-based system
Microservice
Characterization model

ABSTRACT

Service-based architecture is an approach that emerged to overcome software development challenges such as difficulty to scale, low productivity, and strong dependence between elements. Microservice, an architectural style that follows this approach, offers advantages such as scalability, agility, resilience, and reuse. This architectural style has been well accepted and used in industry and has been the target of several academic studies. However, analyzing the state-of-the-art and -practice, we can notice a fuzzy limit when trying to classify and characterize the architecture of service-based systems. Furthermore, it is possible to realize that it is difficult to analyze the trade-offs to make decisions regarding the design and evolution of this kind of system. Some concrete examples of these decisions are related to how big the services should be, how they communicate, and how the data should be divided/shared. Based on this context, we developed the CharM, a model for characterizing the architecture of service-based systems that adopts microservices guidelines. To achieve this goal, we followed the guidelines of the Design Science Research in five iterations, composed of an *ad-hoc* literature review, discussions with experts, two case studies, and a survey. As a contribution, the CharM is an easily understandable model that helps professionals with different profiles to understand, document, and maintain the architecture of service-based systems.

1. Introduction

The service-based approach emerged to overcome common challenges to monolithic software, such as difficulty in maintaining and scaling the software, low productivity, and strong dependence between elements (Richards, 2015; Bogner et al., 2017a). Among the advantages of the service-based approach, we have greater development agility, technological heterogeneity, scalability, resilience, and reuse (Mahmood, 2007; Newman, 2021; innoQ, 2015; Richardson, 2018). Architectural styles such as SOA (Service-Oriented Architecture) (Natis and Schulte, 2003), Self-Contained Systems (innoQ, 2015), and, more recently, microservices (Lewis and Fowler, 2014) follow the service-based approach. These architectural styles are based on the principles of decomposing complex systems into services (basic system units), loosely coupled, and communicating by messages.

Despite the benefits presented, architecting and developing service-based systems is complex. Among the main challenges faced today are the complexity of the development process, difficulty in defining the size and level of coupling of the services, maintaining data consistency, and the necessity to have a robust and automated infrastructure (Mahmood, 2007; Richards, 2015; innoQ, 2015; Newman, 2021; Soldani et al., 2018; Ford, 2018). Furthermore, recent experience reports from companies such as Istio (Mendonça et al., 2021), Segment (InfoQ, 2020), and Uber (Highscalability, 2020) reveal the confusion and highlight the difficulty in designing, understanding, and characterizing the architecture of a given SBS.

The Istio development team thought that microservices would be the right architectural solution for their system. However, since all the components were installed and operated by a single team or individual, they realized that it was not worth paying the price for the greater operational complexity inherent to microservices. Faced with this scenario, the team analyzed the related trade-off, rethought the system, and migrated to

☆ Editor: Laurence Duchien.

* Corresponding author at: Mathematics and Statistics Institute of the University of São Paulo - IME-USP, Matão Street, 1010, Butantã, São Paulo, SP, 05508-090, Brazil.
E-mail addresses: thatiane@ifto.edu.br (T.d.O. Rosa), eduardo.guerra@unibz.it (E.M. Guerra), filipe.correia@fe.up.pt (F.F. Correia), gold@ime.usp.br (A. Goldman).

https://doi.org/10.1016/j.jss.2023.111826
0164-1212/

a monolithic architecture (Mendonça et al., 2021). In the case of Segment, to solve fault isolation and operational overhead problems, the team decided to migrate its monolith to microservices. After three years of migration, the team concluded that microservices were not the most suitable solution for the Segment. Upon realizing this, the team again adopted a monolithic approach more consciously, understanding the real problem to be solved and the advantages and disadvantages of this new migration (InfoQ, 2020). On Uber, one of its teams, which currently adopts the microservices architectural style, reported moving many of its microservices to what they called "macroservices", which would be "well-sized services". One of the justifications for this change is the high complexity of managing thousands of microservices (Highscalability, 2020).

In addition to these issues, when analyzing the discussions presented by Nadareishvili et al. (2016) and Newman (2021), as well as the variety of terms that emerged recently related to service-based architectures, it is possible to notice that there is a fuzzy limit when trying to classify and characterize the architecture of SBSs. Therefore, this scenario demonstrates the value of a solution that characterizes a SBS architecture, since it can help to make grounded design decisions, find acceptable architectural solutions, and know the related trade-off to better meet the desired quality attributes for a system.

By analyzing the state-of-the-art, we identify approaches that mitigate this problem, as they aid in the architectural analysis, recovery, understanding, evaluation, or migration (Bogner et al., 2017b; Granchelli et al., 2017a; Zdun et al., 2017; Engel et al., 2018; Mayer and Weinreich, 2018; Cardarelli et al., 2019; Alshuqayran, 2020; Auer et al., 2021). However, none of these approaches focuses on architectural characterization, nor does it present an empirical assessment of its use and ease of understanding. Faced with this scenario, we elaborated the following main research question (RQ) that guides our work: **RQ** *How to characterize the architecture of SBSs to guide architectural decision making?*

Aligned with the RQ, this work's main goal is *to develop and evaluate a model for characterizing the architecture of service-based systems, adopting microservices guidelines*. During the evaluation step, we aim to validate possible uses of the CharM and evaluate the ease of understanding its structure and the architectural characterization generated from its metrics. Other aspects, such as defining and validating strategies to automate the metrics collection and calculation and verifying the model's ease of adaptation and scalability, are outside the scope of this research.

It is important to clarify that in this research, a characterization model is defined as a way of describing and explaining something, listing some main qualities of the analyzed thing. Therefore, it is useful for identifying problems and solutions in the context where it is applied. The model we developed is named CharM and is organized into four dimensions, allowing us to characterize a SBS's architecture based on static metrics related to the structural attributes of size and coupling. We developed the CharM following the Design Science Research guidelines (Hevner et al., 2004) and observing the ACM SIGSOFT checklist of the standard Engineering Research (Ralph et al., 2022). The Design Science iterations evaluated our model empirically through two case studies and finalized with a broad survey targeted at specialists in SBSs development. In this paper scope, we focus on the CharM description and present the survey results as the final evaluation performed. According to the 58 respondents' assessment, the CharM is easy to understand, and the generated characterization can help understand, maintain, and document the architecture of a SBS. The uses for CharM that were rated most highly by respondents were "understanding the synchronous coupling between services of a system" (93%), "understanding the asynchronous coupling between services of a system" (91%), "evaluating the architecture of a system that is being designed" (91%), and "supporting architectural discussions" (91%).

Thus, our work contributes to the state-of-the-art and practice by proposing an empirically validated theoretical and conceptual model to characterize the architecture of SBSs. Our model aids software architects to better understand the structural characteristics of a given system and, consequently, identify structural aspects that must be improved or maintained. Furthermore, the CharM facilitates the mapping and measurement of different impacts generated in the software architecture. It can also support architectural decisions that consider different quality attributes to balance independence and collaboration of services for a given system.

The rest of the paper is organized as follows. Section 2 explores service-based architectural styles, highlighting microservices presenting definitions and key characteristics, which are fundamental for our model. Section 3 presents the design process of our model, defines what the CharM is, and describes each of its dimensions. Section 4 describes the design and execution of the survey. Section 5 describes the results of the CharM evaluation obtained through the survey. Section 6 address the related work. Section 7 discusses the obtained results. Section 8 marks the conclusions of this paper and presents future works.

2. Service-Based Architectural styles

Inspired by the modular software development (Parnas et al., 1985), the Service-Based Architecture (SBA) style emerged as an approach to overcome challenges common to monolithic style, such as difficulty to maintain and scale the software, low productivity, and strong dependence between elements (Richards, 2015; Bogner et al., 2017a).

The service-based approach composes the Service Oriented Computing (SOC) paradigm, which emerged as a way to develop maintainable and interoperable software (Bogner et al., 2017a). In SOC, services are the "fundamental elements for developing applications/solutions" (Papazoglou, 2003). According to Papazoglou (2003), in this paradigm, services perform from simple functions to complex business processes. Bogner et al. (2017a) explain that services should be "self-contained, composable, technology-neutral, loosely coupled, and allow for location transparency". Richards (2016) indicates that some common characteristics in Service-Based Architectures (SBA) are modularity and distribution.

Architectural styles such as Service-Oriented Architecture (SOA), Self-Contained Systems (SCS), and more recently, microservice (MS) follow the service-based approach. According to Natis and Schulte (2003), Bianco et al. (2007), SOA is a client/server approach where an application is composed of users (clients) and providers (servers) of software services that emphasize the adoption of flexible coupling between components and independent interfaces. Krafzig et al. (2005) explain that SOA designs systems as a network of related services, where each service provides a specified functionality in a well-defined interface. Newman (2015) complements by stating that services collaborate to provide a final set of resources (solution).

Self-Contained Systems consist of an approach that aims to separate the functionalities of a complex system into several autonomous systems, which have their "own user interface, specific business logic, and separate data source". Such autonomous systems communicate/collaborate through RESTful HTTP (Hypertext Transfer Protocol) or lightweight mechanisms, mostly asynchronously (innoQ, 2015).

According to Lewis and Fowler (2014), MS is an approach to designing and developing software as suites of independently deployable small services. The authors also explain that each service is built around business capabilities, runs in its own process, and uses lightweight communication mechanisms. Lewis and Fowler (2014) base their MS definition on nine characteristics that they observe as common in this type of architecture.

These three architectural styles follow concepts such as decomposing complex systems into services (basic system unit), loosely coupled, and communicating by messages. They also share the same principle: dividing complex systems into components which communicate in some way.

There is a series of studies that, in some way, discuss and compare the characteristics of these different service-based approaches, such as the work by Salah et al. (2016), Wolff (2016a,b), Cerny et al. (2017), Rademacher et al. (2017), Shadija et al. (2017), Cerny et al. (2018), Baresi and Garriga (2020), Raj and Sadam (2021). Since the MS is considered one of the most popular service-based architectural styles, we explore it further, presenting its definition and characteristics in Section 2.1.

2.1. Microservices definitions and characteristics

According to Richardson and Smith (2016), this architectural style favors developing software with more agility, scalability, and maintainability. There is still no exact, complete, and consensual definition for the Microservice Architectural Style. Newman (2021) claims that microservices are modeled around a business domain and "are an approach to distributed systems that promote the use of finely grained services that can be changed, deployed, and released independently". The author also highlights that two of the main qualities of microservices are loose coupling and high cohesion (Newman, 2015). Bonér (2016) advocates that in a Microservices-Based Architecture each service must have its own data, and be independent, scalable, and resilient to failure. In one of the explanations presented by Dragoni et al. (2017), MS is defined as "a distributed application where all its modules are microservices". Richardson (2018) explains that a Microservice-Based application is structured "as a collection of loosely coupled, independently deployable services".

Based on an *ad-hoc* literature review, some of the main characteristics of the Microservices Architectural Style are:

- software modularization into independent services (development, testing, deployment, scaling) (Lewis and Fowler, 2014; Jaramillo et al., 2016; Nadareishvili et al., 2016; Dragoni et al., 2017)
- small and isolated services (Lewis and Fowler, 2014; Jaramillo et al., 2016; Nadareishvili et al., 2016)
- services organized around business domain (Lewis and Fowler, 2014; Jaramillo et al., 2016; Nadareishvili et al., 2016; Dragoni et al., 2017; Newman, 2021)
- services with high cohesion (Thönes, 2015; Jaramillo et al., 2016; Nadareishvili et al., 2016; Dragoni et al., 2017; Newman, 2021)
- loose coupling services (Jaramillo et al., 2016; Dragoni et al., 2017; Newman, 2021)
- smart services and dumb pipes (Lewis and Fowler, 2014)
- open, standardized, and lightweight communication mechanisms (Lewis and Fowler, 2014; Nadareishvili et al., 2016)
- decentralized governance and data management (Lewis and Fowler, 2014; Nadareishvili et al., 2016)
- technological heterogeneity (Jaramillo et al., 2016; Nadareishvili et al., 2016)
- "focused on product, not project" (Lewis and Fowler, 2014)
- multi-functional teams (Lewis and Fowler, 2014)
- automated infrastructure (Lewis and Fowler, 2014; Nadareishvili et al., 2016)
- software designed to evolve (Lewis and Fowler, 2014; Dragoni et al., 2017)
- resilient software (Lewis and Fowler, 2014; Dragoni et al., 2017)

After presenting the MS definitions and characteristics most related to this research, in the next section, we describe the proposed model to characterize the architecture of service-based systems.

3. The CharM characterization model

This section presents the design process of the proposed model, explains what the CharM is, and describes its four dimensions.

3.1. Design process

In order to develop a model for characterizing the architecture of SBSs, our study follows the seven guidelines of the Design Science Research (G1: Design as an Artifact; G2: Problem Relevance; G3: Design Evaluation; G4: Research Contributions; G5: Research Rigor; G6: Design as a Search Process; G7: Communication of Research), proposed by Hevner et al. (2004). According to Engström et al. (2020), the Design Science Research (DSR) is a paradigm focused to solve real-world problems and commonly used in research in the engineering and information systems fields. Hevner et al. (2004) explain that the DSR is a problem-solving paradigm, in which kernel theories are "applied, tested, modified, and extended through the experience, creativity, intuition, and problem solving capabilities of the researcher". Furthermore, the authors emphasize that, from the adoption of the DSR, it is possible to iteratively build and evaluate artifacts (constructs, models, methods, or instantiations). As already cited, the artifact developed in this study is a model, which was built and evaluated in five iterations.

The first version of the proposed model emerged from *ad-hoc* bibliographic research (Section 2.1), which was focused on the analysis of MS definitions and characteristics and from discussions with researchers specialized in software architecture. This version of the model is available in a previous paper (Rosa et al., 2020b).

Following this, we extended the *ad-hoc* bibliographic research to identify candidate metrics for each model dimension. It is worth clarifying that, during the CharM design, the main criteria for selecting its metrics were: to be related to the dimensions of the model and the components analyzed, as well as to be extracted independently of technology, source code, or infrastructure. During the second iteration, we still analyzed (trade-off) the influence of a set of microservices patterns on the structural attributes of size and coupling. The results of this trade-off analysis were also published in a previous paper (Rosa et al., 2020a). At the end of this iteration, we obtained a second version of the model with a more concise scope, more refined dimensions, and a list of viable candidate metrics. Version 2 of the model is available in a another previous paper (Rosa et al., 2020c).

In the third iteration, the model was submitted for a first evaluation. We carried out a case study, where we adopted the second version of the model to characterize the architecture of a platform for smart cities, called InterSCity, developed as part of an academic research project. The InterSCity is a smart cities platform created to support the development of collaborative work and to enable experiments in the area. This system

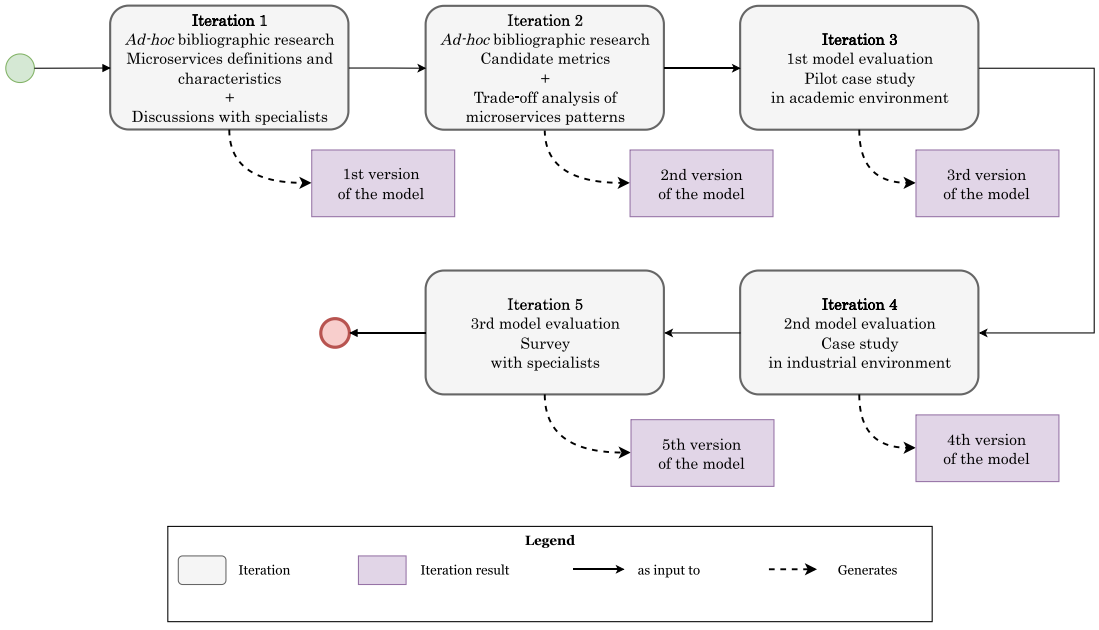


Fig. 1. Design process of the CharM.

has microservices-based architecture and provides a set of high-performance cloud-based mechanisms to manage heterogeneous IoT (Internet of Things) resources, data storage and management, and context-aware resource discovery. This platform is composed of five services (del Esposte, 2018). After applying the model, the architectural characterization was evaluated by 11 project members. We conducted semi-structured interviews and qualitatively analyzed the transcripts using open and axial coding procedures. At the end of the iteration, we obtained feedback regarding the model uses and ease of understanding, as well as collected suggestions for improvements.

We refined the model based on the feedback received during the third iteration. Herewith, we added new metrics, rethought and adjusted the visual presentation of some results' characterization, and optimized the model's explanation. Thus, we generated a third version, called CharM. We submitted this version to an evaluation through a case study in an industrial environment. We carried out this new case study in an online handcrafts marketplace with more than 7 million announced products produced by more than 100,000 active sellers. Since 2018, the architecture of this system is being migrated from the monolithic style to a service-based approach.

In order to make the case study feasible, the CharM was applied to only one part of the system, composed of eight services. After applying the CharM, the generated architectural characterization was evaluated by six development team members. As in the first case study, we conducted semi-structured interviews and qualitatively analyzed the transcripts using open and axial coding procedures. At the end of this iteration, we also obtained feedback regarding the model uses and ease of understanding, as well as collected suggestions for improvements. In order to focus solely on the presentation and evaluation of the final version of CharM, these described case studies (academic and industry) are outside the scope of this paper.

Based on the CharM's evaluation feedback received during the fourth iteration, we identified new uses and improved the results visualization and model explanation. Therefore, we started the fifth and final iteration, in which we evaluate the CharM through a survey, detailed in Section 4. The process design of the CharM is illustrated in Fig. 1.

Given the main goal of this research, it is possible to verify that the built and evaluated artifact is a model, that satisfies the G1 guideline of the DSR. The arguments that we presented in Section 1 demonstrate that it is still difficult to classify and characterize the architecture of SBSs, so we meet the G2 guideline. In this section, we describe the rigor of the artifact (model) design (G5), its development (G6), and evaluation (G3) process. In Section 7, we discuss the research contributions (G4). This paper and the previous ones Rosa et al. (2020b), Rosa et al. (2020a), and Rosa et al. (2020c) communicate our results, which complies with the G7 guideline. Thus, we satisfied the seven DSR guidelines proposed by Hevner et al. (2004).

3.2. Definition

Before defining the CharM, it is important to understand what a *characterization model* is. According to the Cambridge Dictionary (Cambridge University Press, 2022), *characterization* can be defined as "the way in which something is described by stating its main qualities". According to the Oxford Learner's Dictionaries (Oxford University Press, 2022), a *model* can be defined as "a simple description of a system, used for explaining how something works […]". From Hevner et al. (2004) perspective, model-type artifacts "aid problem and solution understanding and frequently represent the connection between problem and solution components enabling exploration of the effects of design decisions and changes in the real world". Thus, we can define a *characterization model* as a way to describe and explain something, stating some of its main qualities. A characterization model also helps to understand the problem of something that is analyzed and consequently identify possible solutions and make conscious decisions.

Therefore, faced with the challenges presented in Section 1, the CharM is a characterization model created to describe and explain the architecture of service-based systems, standing some of its main qualities. From this, the CharM helps professionals identify problems and viable solutions for the architecture of a system and aid the decision-making process. The current version of the CharM focuses on the structural attributes of size and coupling and collects static metrics from the analysis of architectural

design artifacts or APIs (Application Programming Interface) and configurations files.

Since microservices are the current trend for SBSs development (Zimmermann, 2017; Francesco et al., 2017; Bushong et al., 2021; Vera-Rivera et al., 2021), we used some of their guidelines to design the CharM. We extracted five microservices guidelines from the definitions and some characteristics presented in Section 2.1. During the guideline extraction process, we also analyzed definitions and characteristics of modular and other service-based approaches to understand their similarities and differences compared to microservices. Based on this investigation, we identified that microservices, SOA (Service-Oriented Architecture), and other service-based approaches share the same principle: dividing complex systems into smaller components, called services, which communicate in some way.

To define our model dimensions, we considered challenging design decisions in SBSs, especially in the microservices context. Thus, from discussions grounded by the studies of Lewis and Fowler (2014), Hassan and Bahsoon (2016), Bonér (2016), Soldani et al. (2018), Newman (2021), we identified that some of the main microservices challenges are related to the structural attributes of size and coupling (but not limited just to these), such as: defining the services' size, managing distributed data, and defining the services' coupling level. It is worth citing that such challenges are not exclusive to the microservices' context. The studies of Sneed (2006), Mahmood (2007), Perepletchikov et al. (2007) address the importance of size and coupling attributes in SOA and indicate them as challenging. We also identified that patterns such as Business Microservice, Database per Service, Event Sourcing, and Saga aid professionals deal with these challenges (Rosa et al., 2020a) in microservice-based systems. A study by Bogner et al. (2017a), focused on metrics, reinforces the importance of understanding size and coupling characteristics. This study demonstrated that these two structural attributes are among the characteristics most commonly found in the literature related to the maintainability of the SBS. Then, from the analysis of the microservices definitions, characteristics, challenges, and patterns, we extracted the following guidelines that found the CharM: *(i) small and independent services; (ii) loose coupling; (iii) lightweight communication mechanisms; (iv) deployment independence;* and *(v) decentralized data management*.

The CharM can be used to support different stages of the software life cycle, such as its design-time, architecture mapping,[1] and architectural evolution. Besides that, it was created to enable the analysis of different components and perspectives according to the interest scope. Therefore, one of the goals of the CharM is to facilitate the architectural analysis of a system and identify if its characteristics are closer or farther from the listed microservices guidelines. Another goal is to facilitate the identification of the architectural trade-offs and make viable their balancing. To achieve these goals, the CharM is composed of dimensions and metrics that help to verify whether the architecture is adequate to meet the desired non-functional requirements, which, in some way, are influenced by the structural attributes of size and coupling. It is important to clarify that in the current version of the CharM, the metrics of each dimension are collected manually.[2] For this, the professionals can access and explore both architectural design artifacts and (if available) APIs, configuration files, or source code of the system.

It is also worth citing that we designed the CharM to be simple and independent of technology, source code, or infrastructure. Therefore, in its current version, we did not explore some critical features in SBSs, such as cohesion. We chose not to incorporate cohesion in the CharM because we identified that the related metrics (that we have studied so far) tend to make our model more complex and dependent on technology/source code. However, this does not mean the cohesion aspect cannot be incorporated later.

It should also be made clear that the CharM is not a tool-based approach but a theoretical and conceptual model designed to be applied at different software life cycle stages. Besides that, this model could be used as a reference for grounding and developing other solutions with lower levels of abstraction. In the current version of the CharM, we are not interested in guaranteeing the precision and objectivity of the collected metrics nor analyzing data related to technological heterogeneity, performance, and message size.

Furthermore, it is not the CharM's goal to characterize all service-based architectures from the same perspective, ignoring their differences. Nor is it the aim of the CharM to label a given architecture as good or bad or to recommend specific architectural changes. The CharM's goal is to provide information that helps professionals better understand the architecture of a system and support them in making more grounded decisions. It is also important to explain that although the CharM is rooted in microservices guidelines, its application is not limited to systems that follow this architectural style since it analyzes structural attributes considered interesting and challenging in different service-based approaches.

Therefore, we designed the CharM balancing the yearnings for it to be a lean and comprehensive (applicable to different service-based architectural styles) solution. Furthermore, this model was created to be easy to understand, flexible, and expandable to incorporate new dimensions and metrics.

3.3. Dimensions

The four dimensions composing the current version of the CharM are *size, data source coupling, synchronous coupling, and asynchronous coupling*. As illustrated in Fig. 2, each dimension of the CharM was inspired by at least two of the five selected microservices guidelines (Section 3.2). The *Size* dimension is directly related to the guidelines *small and independent services* and *deployment independence*. The guidelines *decentralized data management* and *loose coupling* are present in the *Data Source Coupling* dimension. The *Synchronous Coupling* and *Asynchronous Coupling* dimensions are directly related to the guidelines *loose coupling* and *lightweight communication mechanisms*.

Each dimension is composed of metrics and it is possible choose which components[3] and perspectives[4] will be evaluated. The components that can be analyzed using the CharM are services and modules. In the context of this research, inspired by Richardson (2018), a service is an application with cohesive and well-defined responsibility that implements functionalities related to business tasks. Inspired by Martin (2018), a module is a cohesive set of services deployed together, that is, a deployment unit. Regarding the perspectives, in the CharM context, we can analyze a component in a particular way, *i.e.*, from external,[5] internal[6] or both views.

[1] **Architecture mapping** means identifying the elements that compose the architecture of a system and how these elements relate to each other.

[2] The roadmap that we developed for the manual metrics collection is available at: https://drive.google.com/file/d/1gD-WTV6hGbzTMEd6_NMNX1ud0Nrw-VcI/view?usp=sharing.

[3] **A component** is a part or element of a system. The components analyzed with the CharM are modules and services.

[4] **A perspective** is the way in which a particular system component is analyzed. The CharM permits some external and internal analyses.

[5] **External perspective** pertains to the components outside the context of a system.

[6] **Internal perspective** pertains to the components inside the context of a system.

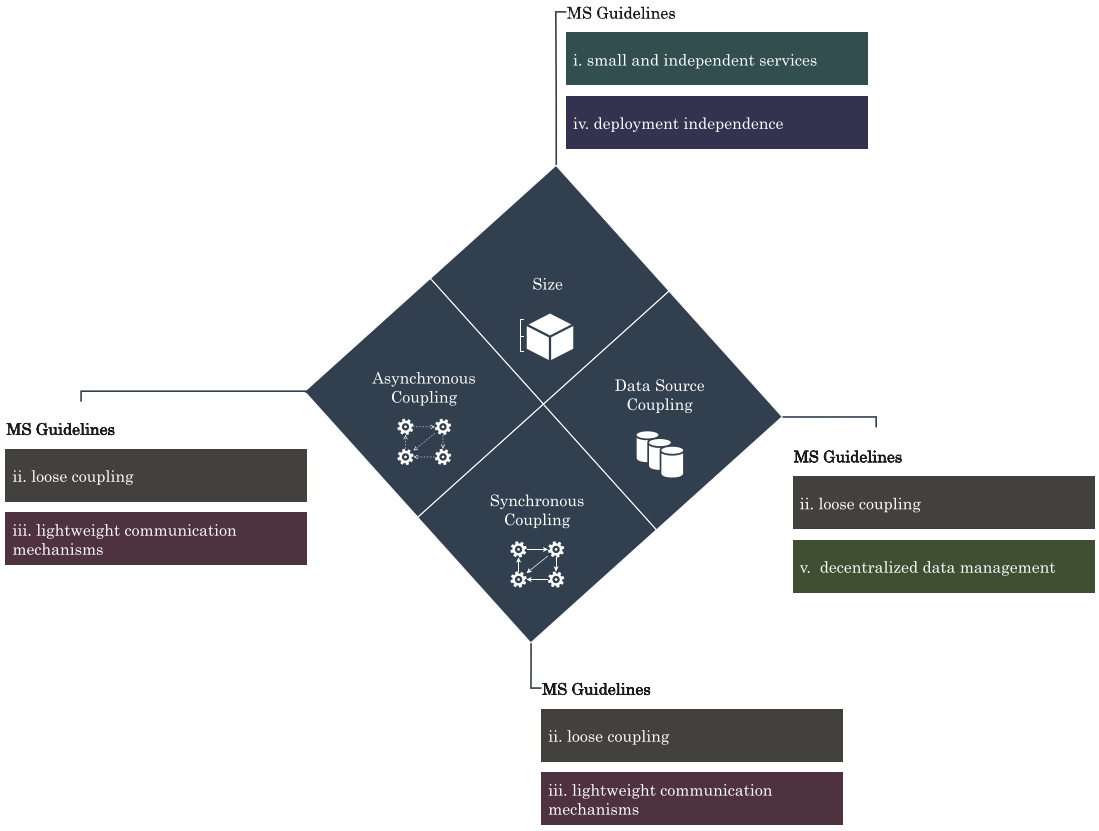


Fig. 2. Relation of the CharM's dimensions and MS guidelines.

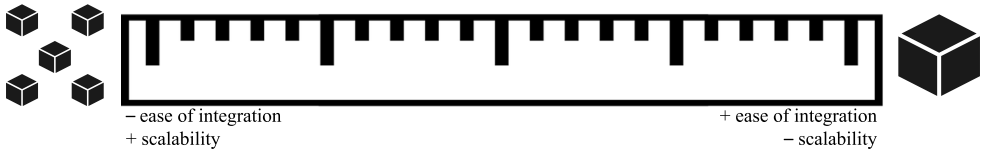


Fig. 3. Ruler of size dimension.

The *Size dimension* aims to characterize the size and composition of different system components and compare them. The components that can be analyzed are services and modules. The metrics that compose this dimension are: *the number of system components*, *the number of services per module*, *the number of operations*[7] *per component*, and *the number of services with deployment dependency*. According to the size of the component, we can obtain different advantages and disadvantages. Therefore, having access to these metrics enables more conscious architectural decisions related to the size of the components. For example, having a single deployment unit, which implements all business functionalities, may facilitate the integration of the components but it may hinder scalability. On the other hand, having each business functionality implemented in a different service, which is an independent deployment unit, may hinder the integration of the components but it may favor the scalability. This scenario is illustrated in Fig. 3.

The goal of the *Data Source Coupling dimension* is to characterize the data source[8] sharing strategy between the components of a system. The components that can be analyzed are services and modules. Besides that, the desired components can be analyzed from external, internal, or both perspectives. The metrics of this dimension are: *the number of system's data sources*, *the number of data sources per component*, *the number of data sources that each component shares with others*, *the number of data sources where each component performs write-only action*, *the number of data sources where each component performs read-only action*, and *the number of data sources where each component performs read and write actions*. Hence, from this set of metrics, we can obtain and evaluate different views of data source coupling between the components of a system and choose a more appropriate sharing strategy. For example, if all system components share a single source, the data coupling degree between the system components will be high, but this strategy may decrease the complexity of managing data consistency. On the other hand, if each component has its own exclusive data source, this may decrease the coupling between system components. However, it may increase the complexity of managing data consistency. Fig. 4 exemplifies this case.

The *Synchronous Coupling dimension* aims to characterize the synchronous interactions between the components of a system. In this research, a synchronous interaction occurs when a component *x* makes a request to a component *y* and waits for a response (inspired by Richardson, 2018). With the CharM, we can analyze the synchronous coupling of services and modules. Moreover, the

[7] **An operation** is a unit of functionality provided by a component (inspired by Shim et al., 2008). In this version of the CharM, an operation is considered a synonym of endpoint.

[8] **Data source** is where it stores the data used by the components of a system

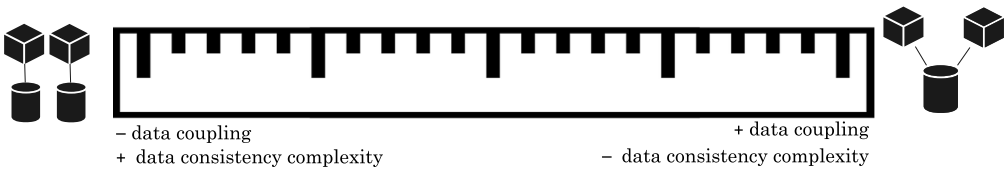


Fig. 4. Ruler of data source coupling dimension.

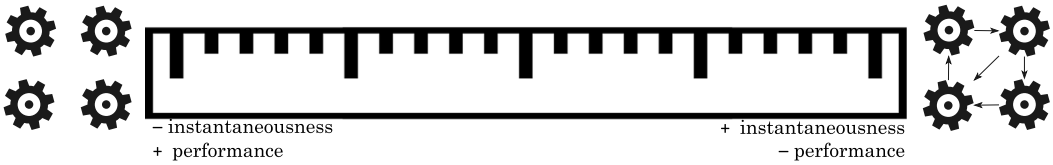


Fig. 5. Ruler of synchronous coupling dimension.

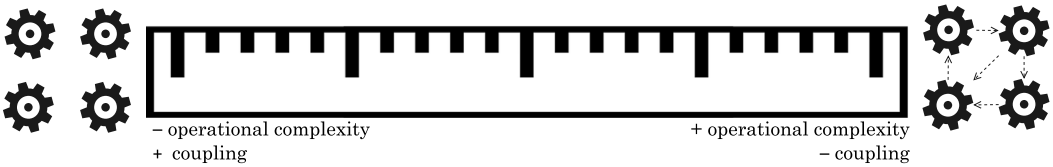


Fig. 6. Ruler of asynchronous coupling dimension.

selected components can be analyzed from external, internal, or both perspectives. The metrics adopted in this dimension are: *the number of clients that invoke the operations of a given component*, *the number of components from which a given component invokes operations*, *the number of different operations invoked by each depending component*, and *the number of different operations invoked from other components*. Therefore, this set of metrics helps to understand the synchronous dependency relationship between the components of a system, as well as the number of operations involved in each interaction. Consequently, it provides relevant information to guide decisions related to synchronous coupling trade-offs. For example, if all services in a system communicate with all the other services synchronously, then message exchange happens almost instantly, but system performance may be impaired. On the other hand, if there is no synchronous interaction between services, then the individual performance of each service may be better, but this strategy may cause a lag in the communication between the services. This situation is illustrated in Fig. 5.

The goal of the *Asynchronous Coupling dimension* is to characterize the asynchronous interactions between the components of a system. Unlike synchronous, an asynchronous interaction occurs when a component x sends a message to a component y and does not wait for a response (inspired by Richardson, 2018). In this research, the main interest is in interactions via message queues. The components that can be analyzed in this dimension are services and modules. Besides that, the selected components can be analyzed from external, internal, or both perspectives. The metrics that compose this dimension are: *number of clients that consume messages published by a given component*, *the number of components from which a given component consumes messages*, *the number of different types of messages consumed by each depending component*, *the number of different types of messages consumed from other components*, *the number of components that consume messages from the queue*, and *the number of components that publish messages in the queue*. Thus, these metrics help to understand the asynchronous interactions between system components, the number of different messages involved in each interaction, and the relationship between components and the message queue. Ergo, they provide relevant information to support decisions related to asynchronous coupling and deal with the trade-offs. For example, if all services in a system communicate with all other services asynchronously, then the operational complexity may increase but, at the same time, it may loosen the coupling. On the other hand, if there is no asynchronous interaction between services, the coupling may be higher, however, operational complexity may decrease. Fig. 6 exemplifies this scenario.

The CharM is an adaptable model, which means that, according to need and interest, the professional can choose which dimensions, metrics, components, and perspectives will be analyzed. Therefore, more valuable metrics may vary depending on the scenario. Thus, one of the main goals of the CharM is to provide professionals with information that allows a better understanding of the system architecture. Besides that, from the generated information, the CharM can help search for an appropriate balance for the software project, not tending to any of the ruler edges. Although there are other relevant dimensions and metrics, the scope of the current version of the model will be limited to those already described. Therefore, the model can be extended to include other dimensions, metrics, and elements in the future. It is also worth citing that it is outside this paper's scope to identify and compare metrics with better results. Complementary, in Appendix A.1, we present a fictitious CharM application scenario, where we illustrate the characterization result, as well as exemplify how our model can be useful to improve and evolve the architecture of a SBS.

4. Survey research design

Considering the research goal, it is fundamental to evaluate the proposed characterization model (CharM), in order to identify the extent of its uses and ease of understanding. As described in Section 3.1, we submitted the CharM to three evaluation stages. In this paper, we present the results of the third evaluation and final, carried out through a survey. Based on this, we defined the following secondary research questions (RQs):

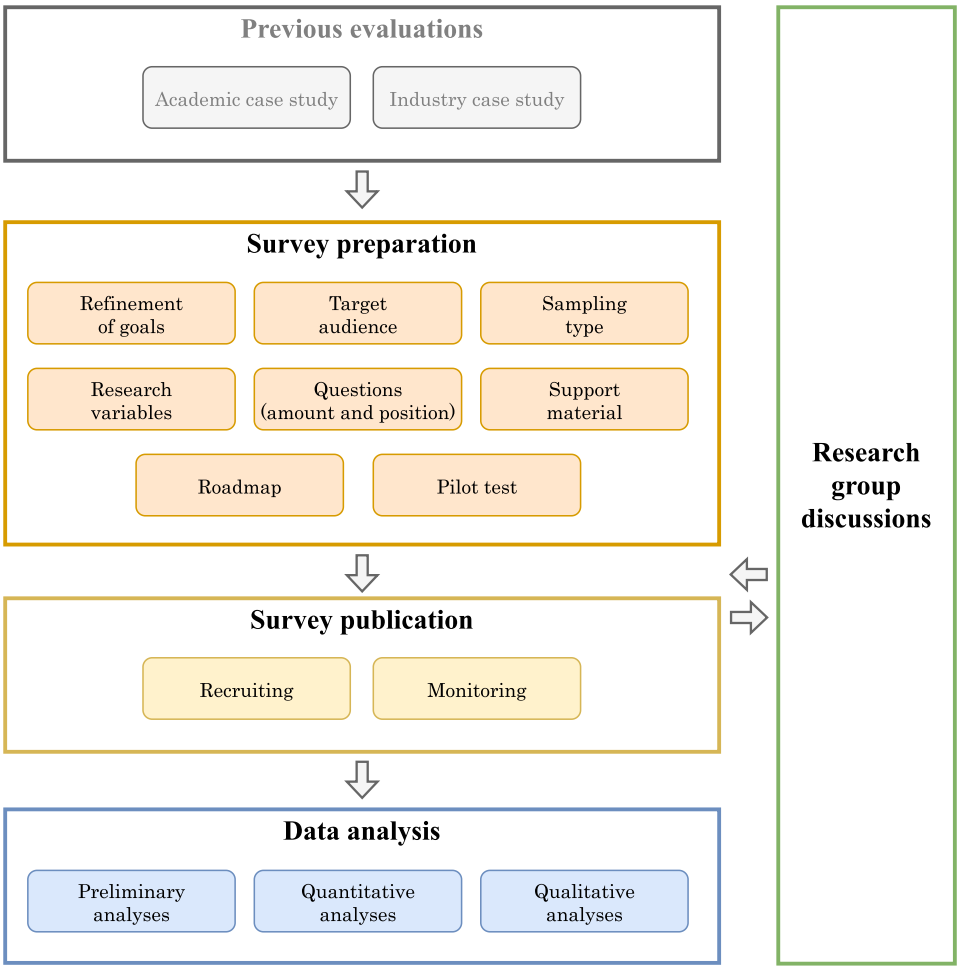


Fig. 7. Overview of the research design of the survey.

- **RQ1:** ***To which extent does the CharM support understanding a service-based architecture?*** — Considerations: we are interested in verifying the usefulness of the CharM for different aspects of understanding the architecture. From identifying and knowing its elements, interactions, and characteristics to the adopted patterns and the related trade-offs.
- **RQ2:** ***To which extent does the CharM support a service-based architecture maintenance?*** — Considerations: we want to analyze the usefulness of the CharM for corrective, evolutionary, and preventive maintenance. Thus, some of the considered points are the identification of adjustments and the quality assessment of the architecture.
- **RQ3:** ***To which extent does the CharM support communicating a service-based architecture to stakeholders?*** — Considerations: we are keen to validate the usefulness of the CharM in disseminating architectural information, helping tasks such as study, explanation, documentation, summarization, and discussion about the architecture.
- **RQ4:** ***How easy is it to understand the CharM?*** — Considerations: we are interested in finding out the level of ease of understanding of the CharM, considering its dimensions and metrics and the results it generates.
- **RQ5:** ***Does the participants' experience influences the perceived usefulness and ease of understanding of the CharM?*** — Considerations: We want to verify if the time of experience with SBA and the level of experience with microservices affect the participants' perception of the different uses of the CharM and the ease of understanding of this model.

In this section, we describe the adopted methodology, the survey sampling and variables, and procedures of execution, replication, and data analysis.

4.1. Methodology

We used an online survey targeting professionals working with service-based systems architecture. Furthermore, the collected data were analyzed quantitatively and qualitatively. Fig. 7 shows an overview of the research design employed in the survey.

As already described (Section 3.1), before the survey, we carried out previous evaluations of the CharM based on a case study in an academic context and another in the industry. The obtained results in these two case studies were used as input to structure the survey.

During the preparation phase, we used materials (Callegaro et al., 2015; Baltes and Ralph, 2020) that enable us to plan and develop a survey with appropriate strategies that was, at the same time, complete, simple and objective. The main tasks performed at this phase were related to the discussions and refinement of goals, delimitation of the target audience that would respond to the questionnaire, and choosing the most appropriate strategy for selecting the population sample (Section 4.2). We also discussed and delimited the independent and dependent variables of the study (Section 4.3), developed the support material (videos and descriptive documents), and defined the questions and roadmap for the survey presentation (Section 4.5). In the end, we carried

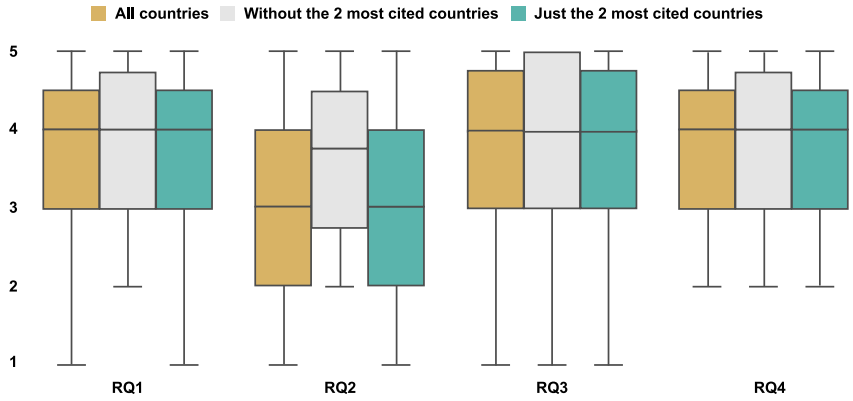


Fig. 8. Relationship between participants' resident country and CharM's evaluation.

out a pilot test round with three professionals and adjusted the survey based on their feedback.

The next phase was the survey publication, which was available for four months. The main activities carried out were recruitment, which involved disseminating the survey to professionals working with service-based systems architecture, and the constant monitoring of the responses received.

As we received answers, we performed some preliminary analyses to identify relevant initial insights and evidence of a state of saturation. After four months, the number of responses stabilized in 58, even with divulgation efforts. Therefore, we move on to the data analysis phase using both a quantitative and a qualitative approach. More details about data analysis are presented in Sections 5 and 4.4.

4.2. Sampling

The sample of participants in this study is non-probabilistic and combines convenience and referral-chain types (Baltes and Ralph, 2020). We adopted the open invitation strategy (Wagner et al., 2020). Thus, the survey was sent out via email to professional contacts in the software development area and working in industry or academia. Furthermore, we shared it on social networks, forums, and discussion lists related to software development and architecture. Recipients were invited to answer the survey and forward it to their peers. In an attempt to obtain the participation of professionals with the desired profile, we present a brief description of the profile of the target population in the research invitation. We adopted these strategies to have a quick send-out and obtain a diverse set of participants. The survey was viewed 446 times (unique clicks) in 29 countries.[9]

In the end, we received 58 valid answers from participants across ten countries. We found that 77% (45) of respondents are concentrated in two countries. Concerned with verifying whether the language background and the questionnaire distribution would affect the survey results, we grouped the evaluation results, considering only these two countries with the highest number of participants (both with the same native language but from different continents). As Fig. 8 illustrates, by ignoring the most frequent countries (gray box), we notice that the data distribution has not changed drastically, maintaining the trend of positive evaluation of the different aspects of the CharM.

[9] These data was extracted from Rebrandly. List of countries where the survey was viewed: Australia, Brazil, Canada, Chile, Colombia, France, Germany, Ghana, Hong Kong, Hungary, India, Italy, Kenya, Mexico, Nepal, Netherlands, Pakistan, Portugal, Romania, Russian Federation, Slovakia, South Korea, Spain, Sweden, Switzerland, United Arab Emirates, United Kingdom, United States of America, and Viet Nam.

4.3. Research variables

Since we want to evaluate the proposed characterization model, we present a list of uses and ask the participants to which extent they consider the CharM to be useful for achieving each one. Furthermore, we wondered how easy it is to understand the model's dimensions, metrics, and results. Therefore, the dependent variables of this study are the *uses* and *ease of understanding of the CharM*. The independent variables are related to the participants' profile, considering the *self-declared time of experience with Service-Based Architecture* and the *self-declared level of experience with microservices*.

The possible uses of the CharM were identified in the two preliminary case studies. The first case study was in an academic environment and the second was in the industry environment. In each case study, we applied the CharM to characterize the architecture of a service-based system. Then, the professionals who worked with the selected systems were invited to analyze the generated results and evaluate the usefulness and ease of understanding of the CharM. In the end, we identified 21 uses for the CharM and organized them into three groups: *(i) understanding of a Service-Based Architecture*, *(ii) maintenance of a Service-Based Architecture*, and *(iii) communication of a Service-Based Architecture to stakeholders*. From this, we extracted the dependent variables of this study.

We suppose that the usefulness and ease of understanding of the CharM could vary according to the user's experience. Therefore, we mapped the respondents' self-declared experience in two aspects. The first independent variable concerns the time of self-declared experience with SBA. In this aspect, we do not use any specific reference or scale. So, we decided to use the following range: (a) *Less than 1 year*, (b) *1 to 3 years*, (c) *4 to 6 years*, (d) *7 to 9 years*, (e) *10 to 12 years*, (f) *13 to 15 years*, and (g) *15 years or more*. The other aspect analyzed was the self-declared level of experience with Microservices. For this, we adopted the five stages of skill acquisition proposed by Dreyfus et al. (1986): *novice*, *advanced beginner*, *competent*, *proficient*, and *expert*. Based on this, in this paper, each level was defined as follows: *novice* - knows the microservice style's theory and principles and applies it in simple contexts; *advanced beginner* - has practical experience of adopting microservice style in real scenarios; *competent* - has experience of adopting microservice style in different real contexts and scenarios; *proficient* - deals with different microservice-based systems and can make decisions in a conscious manner and with a high degree of assertiveness; and *expert* - deals with different microservice-based systems and can make decisions in an assertive way and without significant difficulties. Fig. 9 illustrates the dependent and independent variables of this study.

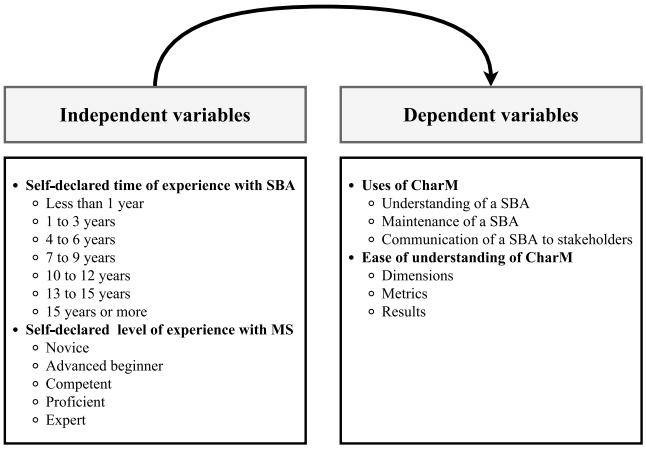


Fig. 9. Research variables.

4.4. Data analysis

At first, we analyzed the participants' profile (Section 5.1) to identify whether we obtained a varied sample. Next, we quantitatively and qualitatively analyzed participants' responses for each of the three CharM use groups (Section 5.2). At that moment, we also tried to identify if any profile was predominant, which, in some way, influenced the evaluation of the use groups. Finally, we also quantitatively and qualitatively analyzed the participants' responses regarding the ease of understanding of the CharM (Section 5.3). Once again, we verified if any profile of participants tended to consider the CharM either easier or more difficult to understand. Quantitative analyses were based on descriptive statistical techniques, such as percentages. During the qualitative analyses, we examined the answers to the open questions. As we analyzed these answers, we organized the data into categories and checked for crossovers and associations. The objective was to identify insights that complement and help us explain the quantitative data. In the final analysis, answers marked as "I have no opinion about this point" were discarded since they did not reveal a positive or negative perception of using the CharM.

4.5. Execution and replication

During this research, a semi-open questionnaire was developed, in which most questions were close-ended. We designed this questionnaire to be complete, simple, and objective. During its elaboration, there were constant discussions among the authors to refine and optimize different aspects of the questionnaire. Furthermore, we submitted it to a test phase to validate, when three professionals with the appropriate profile were invited to answer and evaluate the survey and then present their impressions and suggestions for improvement. The questionnaire was built using Google Forms and was available to the public from 2021-08-14 to 2021-11-30, during which time we received 58 valid responses.

The questionnaire was organized into seven sections. We provided a CharM presentation video[10] explaining its dimensions and metrics in the first section. In the second section, we presented a video[11] that demonstrated the result of the application of the CharM in a fictitious system. In the third section, we collected participants' demographic data. The focus of the fourth section was to collect information on the respondents' professional profile related to academic background, professional performance, and experience with SBA and microservices. Next, we asked about the participants' architectural interests. In the sixth section, we presented a list of 21 uses and asked participants to signalize to which extent the CharM would be useful in each one. In this same section, we asked about the ease of understanding of the CharM's dimensions, metrics, and results. In the last section, the participants could present suggestions for improving the CharM and register general comments.

A replication package is openly available, including the ACM SIGSOFT checklist, the questionnaire, the dataset with the valid answers, and the quantitative analysis script (Rosa et al., 2022).

4.6. Limitations and threats to validity

Internal: By adopting a survey as the CharM evaluation methodology, we limit ourselves to obtaining answers that reveal only the respondents' perceptions, which may not represent the opinion of the total population of professionals who work with service-based architecture. Therefore, to mitigate this threat, we developed a survey based on results that we collected in two previous case studies. This strategy contributed to directing the construction of the survey with really relevant information. Another possible threat is related to the quality of the received answers since participants could need additional explanations to interpret and answer the survey questions adequately. To minimize this risk, we submitted the survey to pilot tests to assess the ease of interpretation of the questions. Furthermore, we presented didactic videos explaining and applying the CharM to contribute to a better understanding of the model and related questions.

External: Considering that the survey was publicly available, using the open invitation strategy, no previous mechanisms were adopted to validate the participants' identities. Thus, we risk obtaining answers from professionals who do not have the profile and knowledge necessary to respond to the questions. To mitigate this threat, we briefly described the target population profile in the survey invitation and collected demographic information to identify if the participants' profile was adequate to answer the survey.

Construct: We adopted protocols for design, execution, and quantitative analysis to facilitate the survey replication. However, the open questions were analyzed qualitatively, which allows for subjective interpretations of some results and discussions.

5. Evaluation results

In this section, we describe the results of the evaluation of the CharM, obtained through a survey. The main points that we explore are participants' profile and the uses of the CharM and its ease of understanding.

5.1. Participants profile

Most participants identified themselves as male (86%), despite the low female participation rate, the numbers are aligned with a recent global census, which evidences that most software development professionals are males (Statista, 2022). We received responses from participants located in different world regions, but most live in Latin America (62%) or Europe (31%). Regarding education level, most participants claimed to have a master's degree (55%) or an undergraduate degree (29%). The participants general profile is presented in Fig. 10.

As illustrated in Fig. 11, most respondents worked in the industry (79%) in the last two years, which is relevant for this research since we can obtain an assessment from a more practical

[10] https://youtu.be/VQRqG9hLBSQ
[11] https://youtu.be/bK9Yg9jmQXY

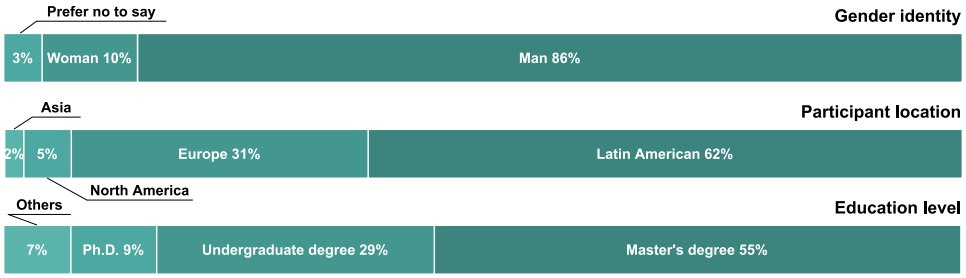


Fig. 10. Summary of participants general profile, according to gender, location, and education level. From top to bottom, we present the (a) gender identify, (b) participant location, and (c) highest completed education level. All percentages were rounded based on simple rules of rounding numbers.

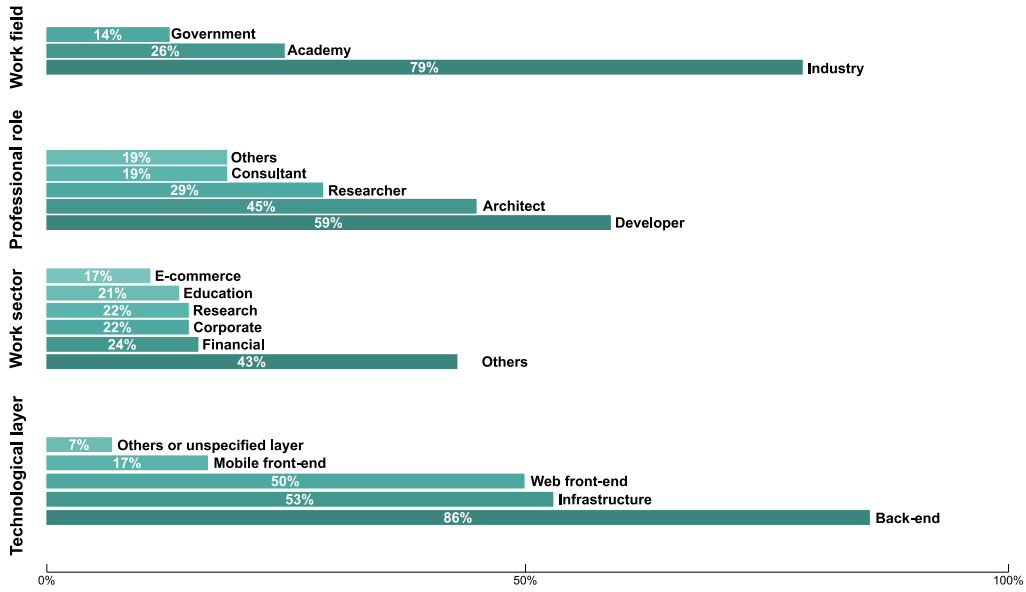


Fig. 11. Summary of participants professional actuation profile. From top to bottom, we present the (a) work field in the past 2 years, (b) professional role in the past 2 years, (c) work sector in the past 2 years, and (d) technological layer in the past 2 years. The sum of percentages exceeds 100% as participants could select multiple answers. All percentages were rounded based on simple rules of rounding numbers.

point of view. Related to the role they perform, most declared to work as a developer (59%) or an architect (45%). The main sectors where survey respondents work are Financial (24%), Corporate (22%), Research (22%), Education (21%), and E-commerce (17%), which demonstrate a good range of sectors, which is also desirable for this research. Participants work in different technological layers, most focusing on the back-end (86%), infrastructure (53%), and web front-end (50%) layers. It is worth mentioning that in these questions, the participants could select multiple answers.

Regarding the self-declared experience with SBA, we received a mixed set of responses from participants with different times of actuation, from less than 1 year to more than 15 years of experience. Respondents also have experience with different service-based architectural styles, where the main are microservices (79%), back-end and front-end service communication (67%), and SOA (64%). In this question, the participants also could select multiple answers. We also obtained a relatively balanced number of respondents with different self-declared experience levels with the Microservices Architectural Style, since 28% declared themselves as a novice, 17% an advanced beginner, 19% a competent, 17% a proficient, and 19% an expert. Therefore, considering the variety of profiles and the number of survey participants, we have a good sample to evaluate the CharM. Fig. 12 summarizes the self-declared experience of the participants with Service-Based Architecture.

5.2. The uses of the CharM

We presented a list of 21 possible uses for our model and asked participants which ones the CharM would be useful for. Based on previous investigation (described in Section 4), these uses were organized into three groups: ***(i) understanding of a Service-Based Architecture; (ii) maintenance of a Service-Based Architecture; and (iii) communication of a Service-Based Architecture to stakeholders***.

As illustrated in Fig. 13, most of the participants who gave their opinion[12] "agree" or "strongly agree" that CharM is useful for ***understanding the architecture of a service-based system***. Using it for *understanding the asynchronous/synchronous coupling* (U7G1 and U9G1) and for *evaluating the architecture of a system that is being designed* (U1G1) stand out, since over 90% of the participants consider that the CharM helps achieve these goals. In the case of uses U1G1 and U9G1, no participant disagreed that the CharM helps achieve them. On the other hand, using the CharM for *identifying adopted architectural patterns* (U2G1) and *identifying the size of services of a system* (U4G1) had the lowest levels of agreement since just over 70% of the participants claimed that the CharM helps achieve these goals. These last two

[12] Participants could choose the option "I have no opinion about this point".

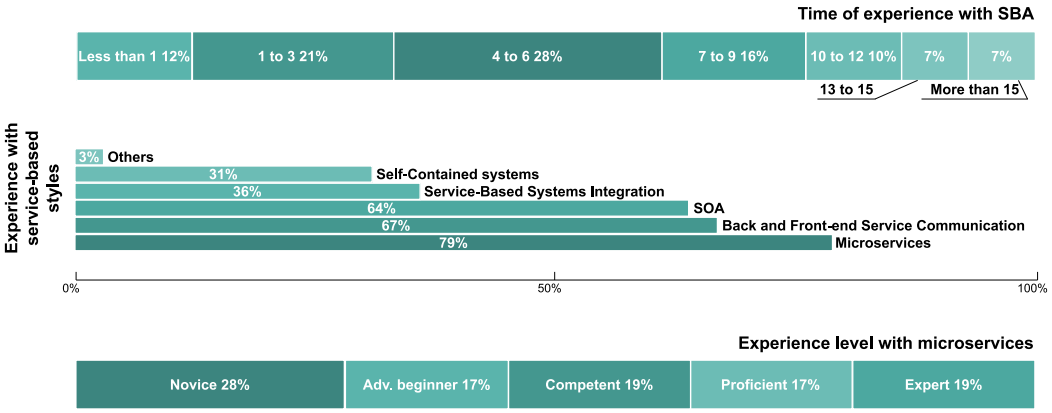


Fig. 12. Summary of participants self-declared experience with SBA. From top to bottom, we present the (a) range of time of experience with Service-Based Architecture, (b) experience with different service-based architectural styles — it was possible to select multiple layers, and (c) experience level with Microservice-Based Architectural Style. All percentages were rounded based on simple rules of rounding numbers.

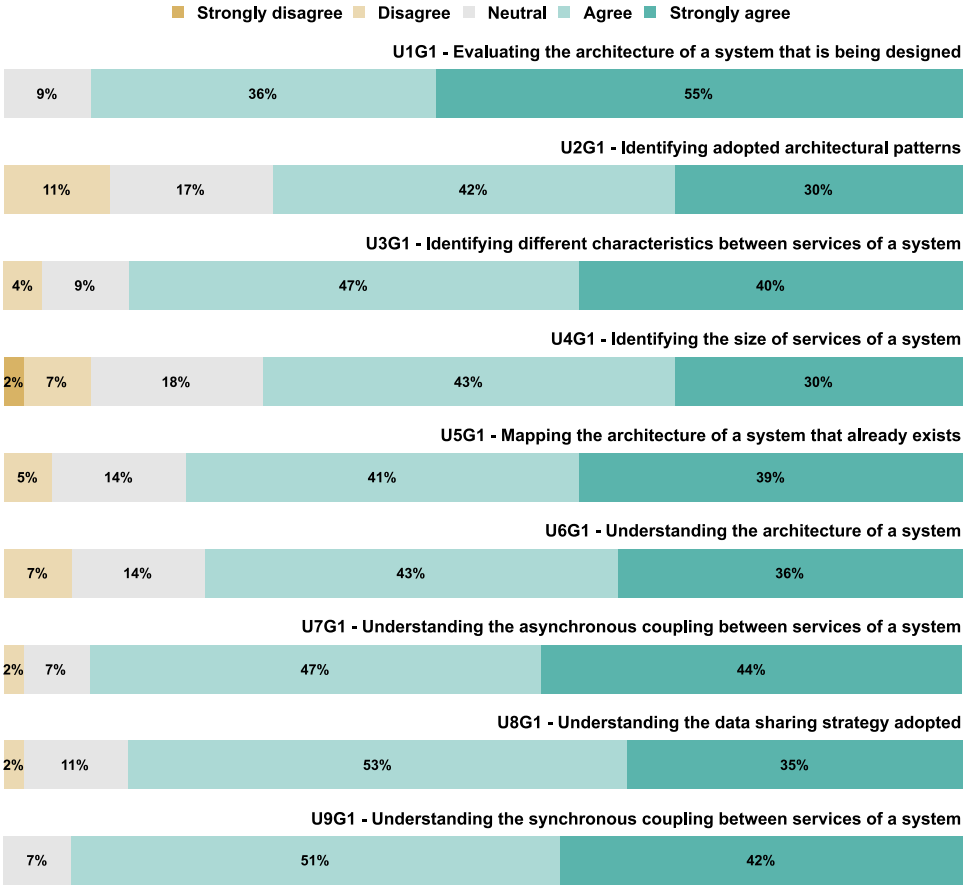


Fig. 13. Participants' perspective to which extent the CharM supports understanding a Service-Based Architecture. All percentages were rounded based on simple rules of rounding numbers.

uses also stood out, since ≈10% of the participants believe that the CharM is not useful to achieve these goals.

Based on the respondents' profile and answers to open questions, we tried to understand what caused this scenario. However, after analyzing the data, no predominant profile was identified. Furthermore, when analyzing the answers to open questions, we did not identify points that could explain why the participants considered the CharM less useful for achieving U2G1 and U4G1 than they did for the other goals.

Analyzing Fig. 14, we can verify that at least 60% of the participants, who expressed their opinion, "agree" or "strongly agree" that CharM can help in some aspects related to ***architectural***

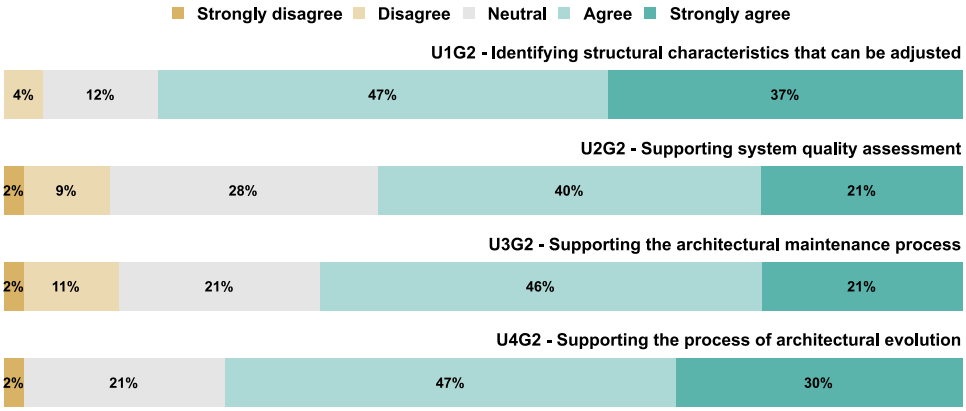


Fig. 14. Participants' perspective to which extent the CharM supports the maintenance of a Service-Based Architecture. All percentages were rounded based on simple rules of rounding numbers.

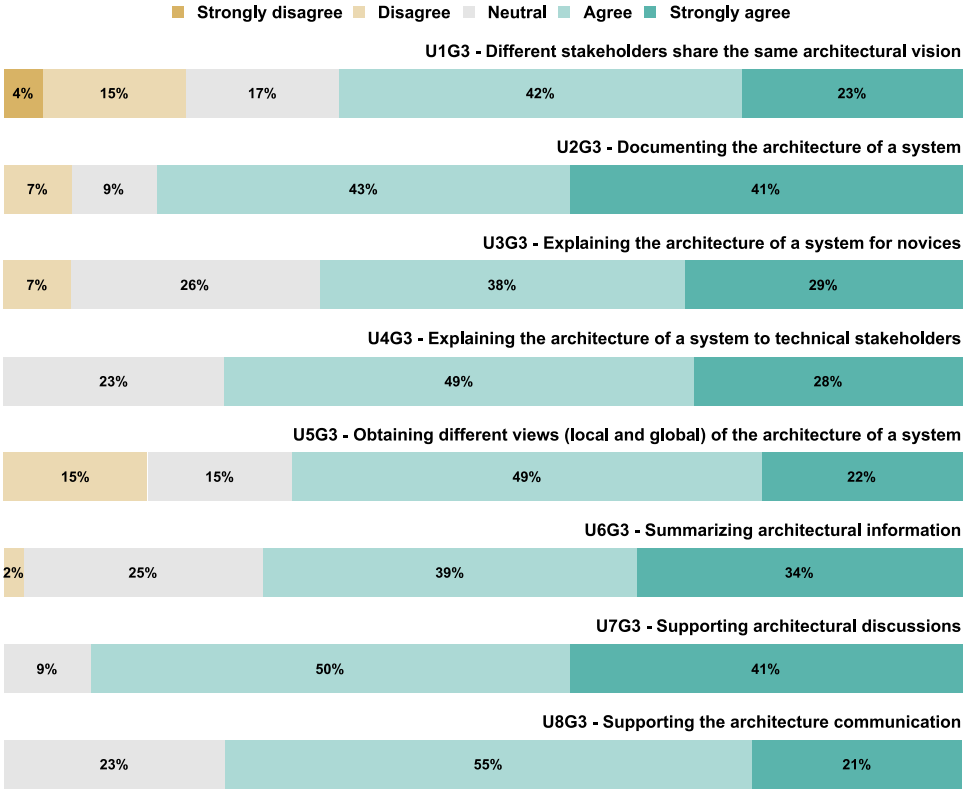


Fig. 15. Participants' perspective to which extent the CharM supports the communication of a service-based architecture to stakeholders. All percentages were rounded based on simple rules of rounding numbers.

maintenance. Its use for *identifying structural characteristics that can be adjusted* (U1G2) stood out in this group, since 84% of participants claimed that the CharM helps achieve this goal. In contrast, at least 11% of respondents disagreed that the CharM is useful for supporting system quality assessment (U2G2) and the architectural maintenance process (U3G2). As in the previous uses group, after analyzing the participants' profile and the open responses, we did not identify evidence that helped understand or explain why a few participants considered the CharM not useful for achieving the objectives U2G2 and U3G2.

As presented in Fig. 15, most of participants, who gave their opinion, "agree" or "strongly agree" that the CharM is useful for ***communicating a Service-Based Architecture to stakeholders***. It is important to highlight that more than 90% of the participants think that the CharM can *supporting architectural discussions* (U7G3). Furthermore, at least 84% of respondents point out that the CharM is useful for *documenting the architecture of a system* (U2G3) and *supporting architectural communication* (U8G3). It is worth mentioning that in the case of uses U4G3, U7G3, and U8G3, no participant disagreed that the CharM helps achieve them. On

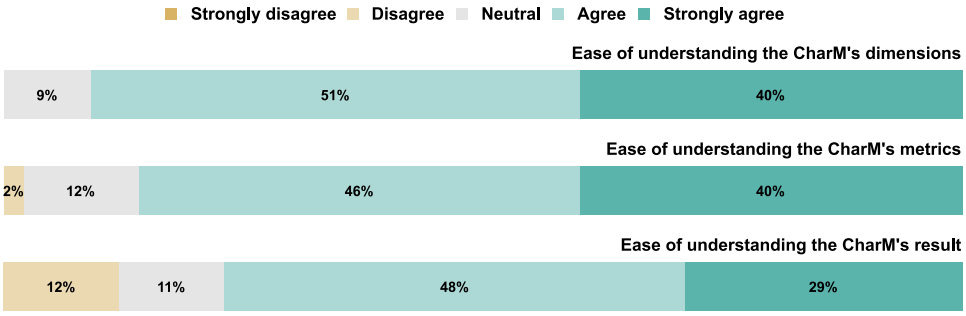


Fig. 16. Participants' perspective on how easy it is to understand the CharM. All percentages were rounded based on simple rules of rounding numbers.

the other hand, almost 20% of participants "disagree" or "strongly disagree" that the CharM can support *different stakeholders to share the same architectural vision* (U1G3). Furthermore, 15% of respondents believe that the CharM is not useful for *obtaining different views (local and global) of the architecture of a system* (U5G3).

Considering the three analyzed groups, the uses U1G3 and U5G3 stood out with the highest rates of participants who disagreed that the CharM can help achieve these goals. Once again, we tried to understand what caused this rate of discordant participants. After analyzing these data, no predominant profile was identified. Furthermore, from the open responses, we did not identify evidence that could indicate why the participants considered the CharM not useful for achieving the objectives U1G3 and U5G3.

During the CharM uses assessment stage, we asked the participants if they identified any new uses, not previously listed in the survey. Four participants contributed, indicating new uses. P9 stated that the CharM is useful for *"supporting the architectural process definition at the project's beginning"*. P11 indicated that the CharM helps *"modernize and migrate strategies for pre-existing workloads"*. P21 reported that the proposed model might be useful in *"decision make before migrating"*. P49 indicated that the CharM might help *"spotting risks related to coupling, performance, and availability"*. Faced with the new uses identified by the participants, we believe that, probably, the CharM can be useful in other scenarios that adopt the same information already explored in the model.

5.3. The ease of understanding of the CharM

After identifying the uses, we asked participants how easy it is to understand the CharM's dimensions and metrics and its results. As illustrated in Fig. 16, 91% of the participants considered it easy to understand the CharM's dimensions, with no participants stating that it is difficult to understand them. When asked about the CharM's metrics, most respondents (86%) also reported that they are easy to understand. Only one participant[13] believed that the CharM's metrics are difficult to understand. Regarding the understanding of the results generated by the CharM, 77% of the participants "agree" or "strongly agree" that they are easy to understand, with only 12% disagreeing.

Some participants presented suggestions for improvements that may contribute to understanding the result generated by the model. Respondent P19 suggested presenting a diagnosis of the result, *"[…] saying whether the architecture is good or not […]"*. Participant P24 stated that it would be interesting and informative to execute *"[…] the software during the presentation"* of the model. P48 claimed that *"[…] it would be great to have a "cheat sheet" or any other material with the summary of the CharM"*. P42 believes that *"[…] it might be helpful to have a kind of guideline that supports the interpretation of results"* generated by the CharM. P30 indicated that it is interesting to clarify *"in which scenarios, it is worth using the CharM"*.

5.4. The use and ease of understanding of the CharM according to professional experience

In order to obtain other perspectives for the CharM evaluation, we explored the results considering the participants' self-declared experience with SBA and with microservices. Participants with 10 to 12 years of experience with SBA tend to agree that the CharM is both useful for understanding the architecture of a SBS (Fig. 17(a)) and that it can support the maintenance process (Fig. 17(b)) and architectural communication with stakeholders (Fig. 17(c)). Furthermore, this same tendency can be verified regarding the proposed model's ease of understanding (Fig. 17(d)).

Analyzing the level of experience with microservices, we can verify that the CharM can be very useful and easily understood by advanced beginners. Participants with this degree of experience with microservices tend to strongly agree that the CharM is easy to understand (Fig. 18(d)) and useful for understanding (Fig. 18(a)), maintaining (Fig. 18(b)), and communicating (Fig. 18(c)) the architecture of service-based systems.

Given the results presented in Figs. 17 and 18, we believe it is relevant to investigate further the perception of the CharM by professionals with different profiles. Furthermore, we are curious to find out whether or not there is some correlation between the independent variables *self-declared time of experience with SBA* and *self-declared level of experience with microservices*.

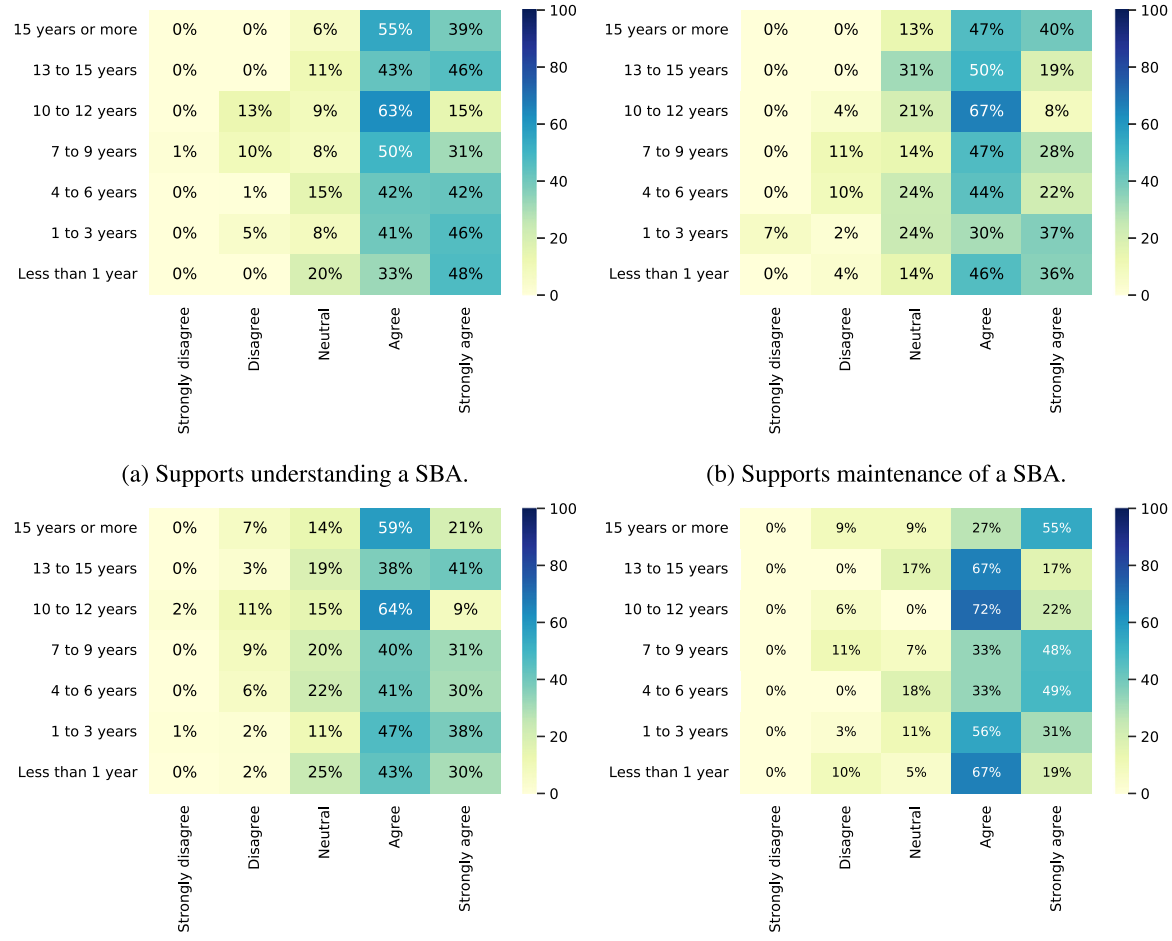
Therefore, considering the survey data, we found that regardless of the experience of the participants, the evaluation of the CharM's usefulness and ease of understanding tends to be positive. That is, the CharM tends to be useful and easy to understand by professionals with different times and levels of experience.

6. Related research

This section presents an overview of scientific studies that propose different approaches which, in some way, support characterizing the architecture of service-based systems. In the end, we compare these approaches and highlight the main differences between them and our study.

The first two studies in the series published by Granchelli et al. (2017a,b) present an approach to assist in recovering and

[13] P22 claimed that he has less than 1 year of experience with Service-Based Architecture and that he is a novice in the microservice architectural style.



(a) Supports understanding a SBA.

(b) Supports maintenance of a SBA.

(c) Supports communication of a SBA to stakeholders.

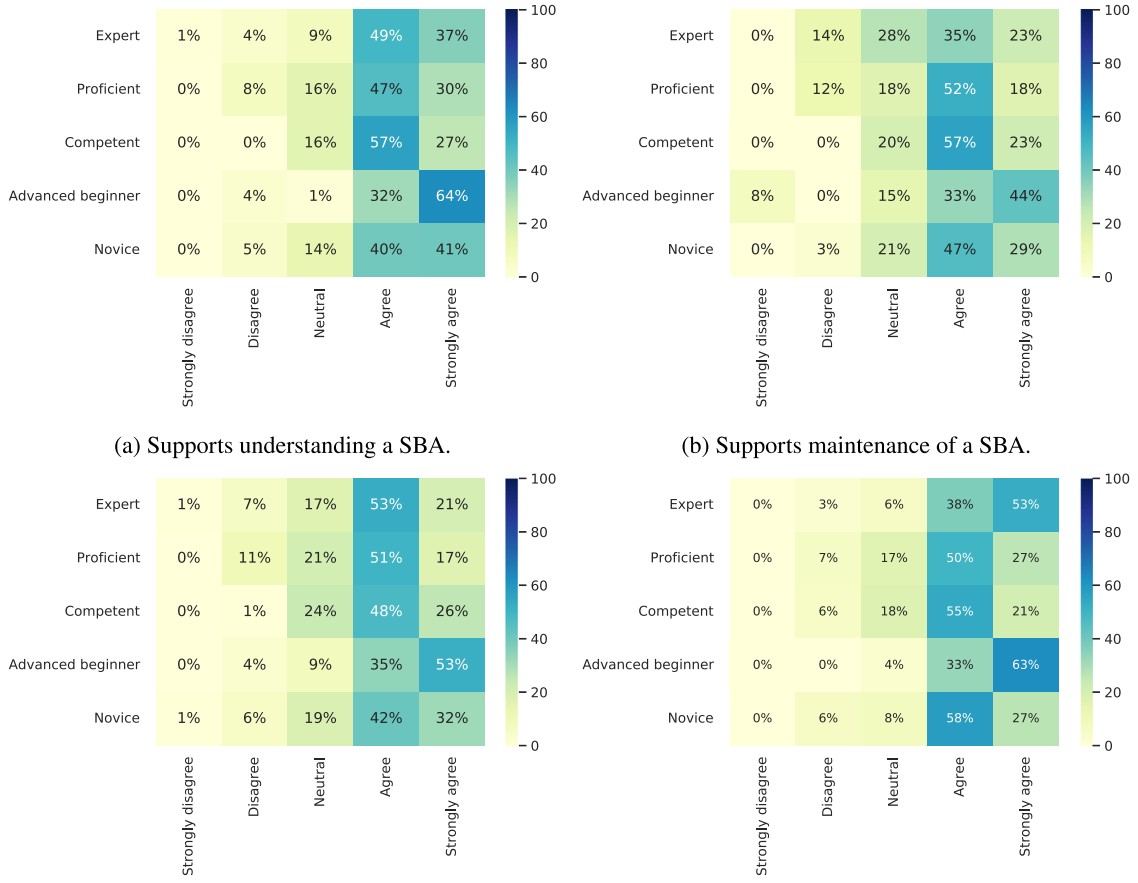
(d) The ease of understanding.

Fig. 17. The CharM's evaluation according to participants' self-declared years of experience with SBA. All percentages were rounded based on simple rules of rounding numbers.

understanding the complexity of microservice-based systems architecture. The last one (Cardarelli et al., 2019) proposes an approach for specification, aggregation, and evaluation of quality attributes for the architecture of microservice-based systems. Such approaches are based on Model-Driven Engineering principles. To validate the proposed approaches, the authors carried out experiments with one open-source system and collected static and dynamic metrics in a semi-automatic way. Some findings of these studies are that the proposed approaches are useful for architectural understanding, documenting, and analyzing, unveiling dependencies between microservices from the development perspective, automatically recovering and measuring architectural models, and continuously assessing the quality of MS.

The study conducted by Engel et al. (2018) presents an approach to evaluate microservice architectures based on principles, such as small size and loose coupling of the services and domain-driven design. To develop the proposed approach, the authors adopted the design of microservice architectures principles extracted from literature review, structured interviews with experts, and the GQM (Goal Question Metric) (Basili and Rombach, 1988) technique to derive the proposed approach's metrics. During the research, a case study was carried out with a project with 50 microservices, in which metrics were collected automatically through a dynamic analysis. Furthermore, discussions were held with project team members. In the end, some identified uses for the approach were to support the identification of hot spots in the architecture of a system, support the design and development of systems, and confirm perceptions of the system's team.

Another series of four scientific studies, led by Bogner, is also related to our work. The first study (Bogner et al., 2017b) proposes a practical maintainability quality model for services- and microservices-based system, named MM4S (Maintainability Model for Services). The study objective is to obtain a simple and practical tool for basic maintainability estimation, control, and specification in the context of services- and microservice-based systems. The authors performed an intra-company focus group combined with a literature review yielding the first requirements for the desired quality model. The second published study (Bogner et al., 2019) in the series aims to calculate service-based maintainability metrics from the runtime data of microservice-based systems. To achieve this goal, they carried out an exploratory study with an example system which was used by six people. The focus of the third study (Bogner et al., 2020a) is to calculate maintainability metrics from machine-readable interface descriptions of RESTful services. An evaluation was performed based on threshold benchmarking with 1737 public RESTful APIs. In the last study (Bogner et al., 2020b), the authors present a continuous assurance method to support the identification and remediation of evolvability-related issues in service-based systems. For the final evaluation of this



(a) Supports understanding a SBA.

(b) Supports maintenance of a SBA.

(c) Supports communication of a SBA to stakeholders.

(d) The ease of understanding.

Fig. 18. The CharM's evaluation according to participants' self-declared level of experience with microservices. All percentages were rounded based on simple rules of rounding numbers.

method, they plan to combine industry case studies with action research. Considering the aforementioned studies, some of the utilities mapped are: maintenance support of service-based systems, evaluation of maintainability of RESTful services, hot spot identification, and early quality evaluation.

The first study (Zdun et al., 2017) of another series, proposes a minimal set of constraints and metrics necessary to evaluate the conformance of an architecture to microservice patterns. To achieve this goal, the authors cataloged an initial set of constraints and metrics. Then, to identify the most significant ones, this initial set was used to manually evaluate 13 architecture models (extracted from the literature). The focus of the second study in the series (Ntentos et al., 2021a) is a method for the semi-automatic detection and resolution of microservice patterns conformance violations. The authors tested this method on a set of 24 models of various degrees of pattern violations and architecture complexity. The last study of the series (Ntentos et al., 2021b) aims to provide foundations for an automated approach for architectural reconstruction, assessing conformance to patterns and practices specific to microservice architectures, and detecting possible violations. To achieve this goal, the authors carried out experiments with a set of 27 models of microservice-based systems from third-party practitioners. Some of the uses of the approach, pointed out by the authors of this series of studies, are: to compare architecture conformance of the current design and possible refactorings, find the root cause of a violation of microservices patterns, measure the quality of microservice decomposition in software architecture models, and suggest possible improvements related to microservice coupling.

Alshuqayran published two studies related to this research. The focus of the first study (Alshuqayran et al., 2018) is to present an approach based on Model-Driven Engineering (MDE) to recover the architecture of microservice-based systems. This approach was evaluated through empirical studies that analyzed eight microservice open-source projects. In the second publication (Alshuqayran, 2020), the author explains that the objective of the proposed approach is to comprehend the complexities of MS by developing a bottom-up reverse engineering process. The evaluation of the approach proposed in the first study was complemented with an empirical study on nine open-source microservice-based projects and a case study with a large open-source microservice-based system. Some of the listed uses for the approach are: to create architectural documentation, obtain performance diagnostics at the container level, identify several concepts and support the definition of the underlying features and behavior of microservice-based systems, and help identify and build the relations between services.

The research conducted by Mayer and Weinreich (2018) presented an approach that aims to extract and analyze the architecture of a microservice-based system based on a combination of static service information with infrastructure-related and aggregated runtime information. Such an approach is based on three levels in a microservice-based system: service, infrastructure,

Study	Support understanding a SBA	Support the maintenance of a SBA	Support the communication of a SBA to stakeholders	Independent of code or infrastructure?
Granchelli et al. (2017a,b); Cardarelli et al. (2019)	Yes	Yes	Yes	No
Engel et al. (2018)	Yes	Yes	Yes	No
Bogner et al. (2017, 2019, 2020a,b)	Yes	Yes	No	No
Zdun et al. (2017); Ntentos et al. (2021a,b)	Yes	Yes	No	Yes
Alshuqayran et al. (2018); Alshuqayran (2020)	Yes	Yes	Yes	No
Mayer and Weinreich (2018)	Yes	Yes	Yes	No
Auer et al. (2021)	No	Yes	Yes	Yes
This study	**Yes**	**Yes**	**Yes**	**Yes**

Fig. 19. Comparison of related research.

and interaction. The proposed approach was evaluated through a survey and interview study with 15 architects, developers, and operations experts. In the end, some of the uses identified for the approach were to provide a high-level overview of the system, analyze architecture evolution, provide an overview of a service's functionality, obtain automatic documentation of a MS, and determine which microservices and their functions will be affected by changes.

The work by Auer et al. (2021) presents a framework for supporting companies in discussing and analyzing the potential benefits and drawbacks of the migration and re-architecting process. During the study, a survey was carried out in the form of interviews with 52 professionals. The main use of the framework, pointed out by the authors, is to help companies avoid architectural to microservices migration if it is not necessary, especially when they might get better results by refactoring their monolithic system or re-structuring their internal organization.

Fig. 19 compares our study with the related papers described in this section. From the investigation carried out to date, we have not identified approaches with the same objective as the CharM. That is, to characterize the architecture of service-based systems based on microservices guidelines and static metrics of the structural attributes of size and coupling. Although the analyzed approaches do not have the same objective as the CharM, they have some uses in common. Therefore, they can support the maintenance, understanding, or communication of the architecture of service-based systems. Thus, when analyzing the related studies,[14] we can observe that all the other presented approaches, in some way, support the maintenance of the architecture of service-based systems. Nevertheless, we did not clearly identify that the approach proposed by Auer et al. (2021) supports understanding a service-based architecture and that the approaches presented in the series of studies conducted by Bogner et al., Zdun et al. and Ntentos et al. support the communication of a service-based architecture to stakeholders.

Another feature we observed when comparing our research with the others was whether the proposed approach's analysis depends on the software source code, technology, or infrastructure. We found that only our model and the approaches proposed by Zdun et al., Ntentos et al. and Auer et al. have this independence. This feature makes the CharM could be used at different software life cycle stages.

Furthermore, although the proposed approaches were evaluated empirically, we found that the use and ease of understanding aspects were not the analysis focal point in any of the related work. That is, none of the studies that we analyzed have deeply investigated how useful the proposed approaches are for certain tasks or how easy these approaches are to understand. In contrast, one of the focuses of our work is to empirically evaluate the uses of CharM and its ease of understanding.

7. Discussion

As explored in Section 6, there are approaches focused on the analysis, recovery, and understanding of the architecture of service-based systems (Granchelli et al., 2017a; Engel et al., 2018; Mayer and Weinreich, 2018; Alshuqayran, 2020), others are more focused on the evaluation of quality attributes (Cardarelli et al., 2019; Bogner et al., 2017b) and to the analysis of conformance to architectural patterns (Zdun et al., 2017), and there are others that support the architectural migration process (Auer et al., 2021). Despite this, experience reports from companies such as Istio (Mendonça et al., 2021), Segment (InfoQ, 2020), and Uber (Highscalability, 2020) evidence the difficulty in characterizing and defining which architectural structure is the most suitable for a given system. Furthermore, based on the work by Nadareishvili et al. (2016) and Newman (2021) and the variety of terms such as nanoservice, microservice, and macroservice, we can notice that there is a fuzzy limit when trying to classify and characterize the architecture of SBSs. Such a scenario is an indication that it is still necessary to invest in the development of approaches for evaluating, characterizing, and guiding the architecture of SBSs.

That said, the CharM is being developed to help identify structural characteristics (size and coupling), which allows us to describe and understand the architecture of SBSs, as well as their components. It is important to clarify that, different from approaches such as MicroART (Granchelli et al., 2017a), MAAT (Engel et al., 2018), MicroQuality (Cardarelli et al., 2019), MiSAR (Alshuqayran, 2020), and RAMA (Bogner et al., 2020a), the CharM is not a tool-based approach but a theoretical and conceptual model designed to be applied at different software life cycle stages. Hence, the CharM combines some of the advantages of the approaches presented in Section 6 for architectural evaluation, recovery, and evolution. As already demonstrated, this study differs from those cited since it evaluates to which extent our approach (CharM) is useful for different purposes and is easy to understand.

Besides that, it is worth mentioning that we selected the microservices guidelines that underlie CharM through *ad-hoc* bibliographic research. Therefore, we did not adopt a strict protocol for mapping the microservices definitions and characteristics.

[14] The summary of the related studies analysis is available at the replication package (Rosa et al., 2022).

Despite this, the entire process of designing the CharM was grounded on the seven Design Science Research guidelines. We also clarify that adopting the current version of the CharM in large-scale projects is unfeasible since the metrics are collected manually. Cardarelli et al. (2019) and Ntentos et al. (2021a) highlighted this same challenge of evaluating their approaches in large-scale systems.

Considering this scenario and from the analysis of survey results (Section 5), we could answer the five research questions specified in Section 4.

RQ1: *To which extent does the CharM support understanding a service-based architecture?* Granchelli et al. (2017b) claim it is challenging to have a clear architectural overview of a microservice-based system. However, Mayer and Weinreich (2018) argue that it is essential to understand how a system is divided into components and how they interact. In addition, Bass et al. (2012) explain that by understanding the architecture of a system, it is also possible to understand the ability of its architecture to meet the quality attributes. Faced with this importance and challenge, the CharM was assessed as an approach that contributes to understanding the architecture of a service-based system, as well as the approaches proposed by Mayer and Weinreich (2018), Cardarelli et al. (2019), Alshuqayran (2020). Especially, the survey participants believe that the CharM supports understanding the synchronous and asynchronous couplings between services, as well as is useful for evaluating the architecture of a system that is still being designed. This participants' perception is in line with some of the CharM objectives. Nevertheless, although there is a dimension of the CharM focused on the structural attribute of size, the survey respondents did not indicate the identification of the size of the components as one of the main uses of the CharM. This result may be related to the discussion of the subjectivity of defining the service's size, as explored by Newman (2021).

RQ2: *To which extent does the CharM support a service-based architecture maintenance?* Bogner et al. (2017b) explain that the maintainability, that is, the degree of effectiveness and efficiency in modifying, correcting, improving, extending, or adapting software is fundamental for organizations. Based on the survey participants' answers, the CharM can contribute to the architectural evolution of SBSs, especially to identify structural characteristics that can be adjusted. But, different from the approach MM4S (Bogner et al., 2017b), system quality assessment is not among the main uses of the CharM (even with a positive acceptance). A possible explanation for this perception from some participants is that the results generated from the CharM do not show an explicit relationship with the non-functional requirements nor improvement suggestions, as in the approach proposed by Ntentos et al. (2021b). This interpretation is reinforced by the following comment from participant P54, who suggested *"modeling the data collected by the CharM to recommend architectural improvements according to the context and objective of each service-based system"*.

RQ3: *To which extent does the CharM support communicating a service-based architecture to stakeholders?* For Bass et al. (2012), the architecture of a system must be communicated clearly and unambiguously to all stakeholders. Considering this, the survey participants believe that the CharM can support the process of communicating service-based architecture to stakeholders, especially in architectural discussions and documentation. This participants' perception demonstrates that the CharM fulfills the objective of facilitating the analysis of the architecture of a service-based system, similar to the MicroART approach (Granchelli et al., 2017a). Although the CharM allows choosing the system's components and perspectives that will be analyzed, a small index (15%) of survey participants did not perceive the model's utility for obtaining different views of the architecture of a system. Therefore, we believe this may be a CharM use that can be better explored and explained in future versions.

RQ4: *How easy is it to understand the CharM?* The ease of understanding is an important aspect to be evaluated in new artifacts, as it helps determine their ease of use (Lederer et al., 2000) and can influence their perceived usefulness (Davis, 1989). In general, the survey participants indicated that the CharM is easy to understand. However, considering the analyzed aspects (dimensions, metrics, and results), the respondents indicated that they faced some difficulty understanding the generated results. We believe this difficulty is justified by the volume of data presented and the need to understand the analyzed system and relate all the metrics and perspectives presented. As already mentioned, none of the approaches analyzed in Section 6 has undergone an assessment of ease of understanding or use.

RQ5: *Does the participants' experience influences the perceived usefulness and ease of understanding of the CharM?* We hypothesized that participants with different profiles would have different use perceptions of the CharM and that participants with less time and a low level of experience would tend to have more difficulty understanding the model. However, we found that the participants' profile did not significantly influence (negatively) their perception of usefulness and ease of understanding of the CharM. Despite this, we believe it is relevant to investigate further the perception of CharM according to professional experience since this type of analysis can help better understand the results of the CharM's uses and identify improvements to be implemented. Some approaches analyzed in Section 6, such as those proposed by Mayer and Weinreich (2018) and Auer et al. (2021), were also evaluated through a survey. However, the authors did not investigate whether the perception of their approaches varied according to the participants' profiles. Auer et al.'s study only accepted experienced respondents without an academic background.

Given the positive evaluation of the CharM obtained from the survey, we strengthen the evidence that this model is useful to characterize the architecture of service-based systems, regardless of time and level of professional experience. Despite this, there are aspects of the CharM to be explored and improved. These aspects are mainly related to dimensions and metrics (type and collection type), presentation and explanation strategy, and the possibility of the model integration with existing technologies. To implement some of these improvements, we are studying strategies for collecting and analyzing metrics semi-automatically, as in the Mayer and Weinreich (2018), Cardarelli et al. (2019), Bogner et al. (2020b), and Alshuqayran (2020) studies. In addition, we are also working on optimizing the generation of the views (rulers, graphs, and diagrams) of the CharM metrics (Santana et al., 2021).

8. Conclusion

Faced with the challenge of characterizing the architecture of service-based systems, we sought a way to mitigate this problem and, at the same time, support the architectural decision-making process. Therefore, the main objective of this paper was to develop and evaluate a model to characterize the architecture of SBSs, adopting microservices guidelines. We called this model CharM.

To achieve this goal, we followed the seven Design Science Research guidelines, which allowed us to build and evaluate the CharM iteratively and incrementally, based on *ad-hoc* literature review, discussions with software architecture experts, case studies, and a survey. Thus, this paper defined the CharM and described its dimensions and metrics. In addition, we presented

the result of this model's evaluation through a survey, answered by 58 professionals who work with the architecture of SBSs.

During the evaluation stages, we identified that the characterization generated by the CharM is useful for professionals with different profiles (RQ5), helping them to understand the architecture of a SBS and in the process of architectural maintenance and communication. The survey results evidenced that our model is useful for understanding the synchronous and asynchronous couplings between services and evaluating a SBS's architecture at design time (RQ1). Furthermore, another use of the CharM that stood out was the possibility of identifying structural characteristics, which can be adjusted (RQ2). The characterization generated by the CharM was also highlighted as useful to support discussions and compose architectural documentation (RQ3). It is also worth mentioning that the CharM had a mostly positive evaluation since it was considered useful for the 21 uses presented in the survey. The survey results also showed that the CharM is easy to understand (RQ4).

Thus, when considering the main research question, RQ *"How to characterize the architecture of SBSs to guide architectural decision making?"*, we believe that the results presented in this paper demonstrate that the CharM is a useful model to characterize the architecture of SBSs and that it supports architectural decision making. Therefore, the main contribution of this work is the CharM, which is a valuable resource that helps professionals with different profiles to understand, document, and maintain the SBSs' architecture. Despite the positive result of the model evaluation, some improvements can be implemented, such as: not limiting the CharM dimensions to the structural attributes of size and coupling, exploring other types of metrics (static and dynamic), adopting strategies for semi-automatic collection of metrics as well as the automatic generation of views, and presenting results that relate explicitly to the influence of the architecture characteristics on the quality attributes.

As future work, we are planning to develop automation tools related to collecting metrics from the model, generating views (rulers, graphs, and diagrams) of the characterization of the architecture of the analyzed system, and identifying architectural patterns from the metrics of the model. We also intend to evolve the model incorporating new dimensions and metrics related to different attributes, test the resource of analysis profiles, and submit it to new iterations of evaluation in an industrial environment. We also plan to investigate which metrics perform best and add the most value to our model. During the new evaluation stages, we plan to ponder the flexibility of our model, as well as examine the correlation between the current dimensions and identify possible impacts of the addition of new dimensions or metrics. Furthermore, we want to develop a catalog of evolutions, which, combined with the CharM, serve as a guide to adjust the architecture according to the desired architectural approach.

CRediT authorship contribution statement

Thatiane de Oliveira Rosa: Conceptualization, Methodology, Software, Investigation, Data curation, Writing – original draft, Visualization. **Eduardo Martins Guerra:** Conceptualization, Methodology, Writing – review & editing. **Filipe Figueiredo Correia:** Conceptualization, Methodology, Validation, Writing – review & editing, Supervision. **Alfredo Goldman:** Conceptualization, Methodology, Writing – review & editing, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All the links to access the research data are available in the paper.

Acknowledgments

We would like to thank all professionals who accepted our invitation and donated their time to evaluate the CharM and contribute to our research. We thank João Francisco Lino Daniel for very useful discussions on the topic of this paper and Bruno S.C.M. Vilar for the support during quantitative analysis. We also would like to thank the FAPESP (São Paulo Research Foundation) for their research support and the CNPq (National Council for Scientific and Technological Development) for financial support Grant #308996/2022-4.

Appendix. Appendices

A.1. Fictional scenario to demonstrate charm application

To demonstrate the application of the CharM dimensions and metrics, we created a fictional system named Pingr. A more detailed version of this demo scenario is available on YouTube.[15] The Pingr is a social network in the microblog format, where users can make posts limited to 160 characters. Each post is called *ping* and can receive *likes* and *pongs*. A *pong* means that other users shared a *ping*. The historical set of *pings* is displayed in a post feed, called *main table*. To post *pings*, people need to create an account. After creating an account, a user can post *pings* and follow other users. Users can also interact privately by chat. The quality requirements of Pingr are availability, scalability, performance, failure resilience, security, and privacy. Fig. 20 presents an overview of the Pingr architecture.

As illustrated in Fig. 20, there are four internal services, represented by the green circles. These services are *User*, *Chat*, *Ping*, and *Feed*. There is also one external service responsible for some authentication tasks. This external service is the gray circle. The white rectangles around the circles represent the modules, that is, deployment units. The yellow cylinders represent the data sources. The continuous blue lines represent the synchronous interactions between the services. The red dotted lines represent the asynchronous interactions between the services. Cardinalities with a blue background represent the number of operations involved in synchronous interactions. Cardinalities with a red background represent the number of messages involved in asynchronous interactions.

A.1.1. Services characterization based on charm

The **User Service** is the biggest service in the Pingr system, with nine operations. It accesses one exclusive data source performing read and write actions. Related to synchronous coupling, it is the service with the biggest internal importance and is the only one with external dependence. It is one of the services that perform asynchronous interactions, publishing three topics on the message queue that the Chat Service consumes. Fig. 21 illustrates the User Service characterization, based on CharM dimensions and metrics.

The **Ping Service** is the second-biggest service in the Pingr system, with five operations. It accesses one data source, which it shares with the Feed Service, performing read and write actions.

[15] Demonstration of application of the CharM: https://youtu.be/bK9Yg9jmQXY

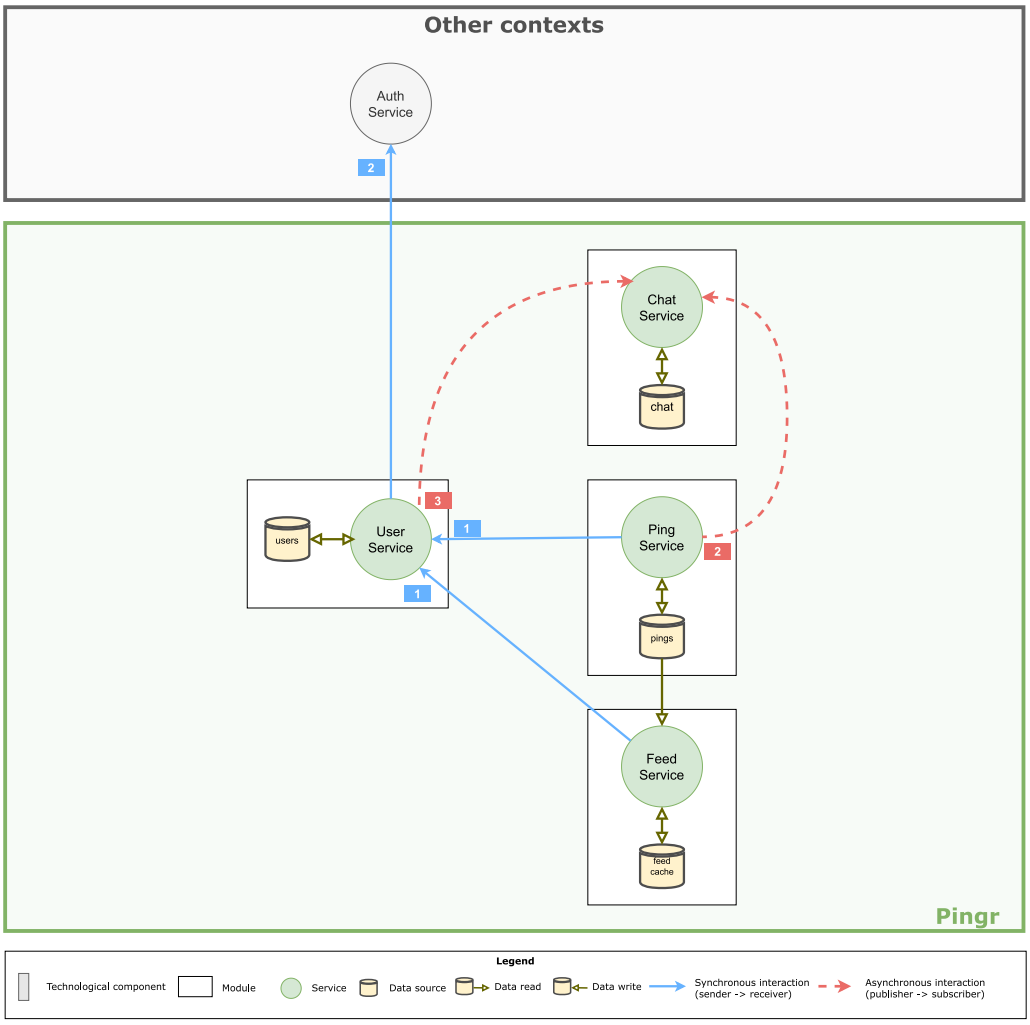


Fig. 20. Pingr architecture.

It has a synchronous dependency on User Service. It is one of the services that perform asynchronous interactions, publishing two topics on the message queue that the Chat Service consumes.

The **Chat Service** has three operations. Access one exclusive data source performing read and write actions. It does not have synchronous coupling. It is the service with the biggest asynchronous dependency, consuming topics published by User Service and Ping Service.

The **Feed Service** is the smallest service in the Pingr system, with two operations. It accesses two data sources, one exclusive and one shared with the Ping Service. It has a synchronous dependency on User Service. It does not have asynchronous coupling.

A.1.2. System characterization based on charm

The Pingr is a service-oriented system composed of four services. Each module is composed of a single service. User Service is the biggest, with nine operations, while the smallest service (Feed Service) has only two operations. The system has four data sources, one of which is shared between two internal services. 60% of interactions between services are synchronous. On the other hand, 40% of interactions between services are asynchronous. Fig. 22 illustrates the characterization of the Pingr system, based on CharM dimensions and metrics.

A.1.3. Example of how the charm can be useful to improve the pingr architecture

As presented, the main uses of the CharM, perceived by the survey participants, are understanding the synchronous and asynchronous coupling between services of a system, evaluating the architecture of a system that is being designed, and supporting architectural discussions. Therefore, in this section, we are going to explore these uses in the context of the Pingr system to exemplify how our model can be useful in improving and evolving the architecture of a system.

Given that Pingr is a fictitious system, we can consider it to be at design time. Thus, the characterization generated from the CharM could be very valuable in assessing whether the designed architecture is adequate to meet the pre-established quality requirements. Furthermore, the metrics, rulers, and graphs generated could guide and support reflections and discussions about this architecture.

The characterization also demonstrates that most of the Pingr services interact synchronously. This characteristic can directly influence the system's performance. Since performance is a quality attribute listed as critical in Pingr, it is worth reflecting on whether it is advantageous to transform some synchronous interactions into asynchronous ones. For example, the coupling between the services Ping and User and Feed and User.

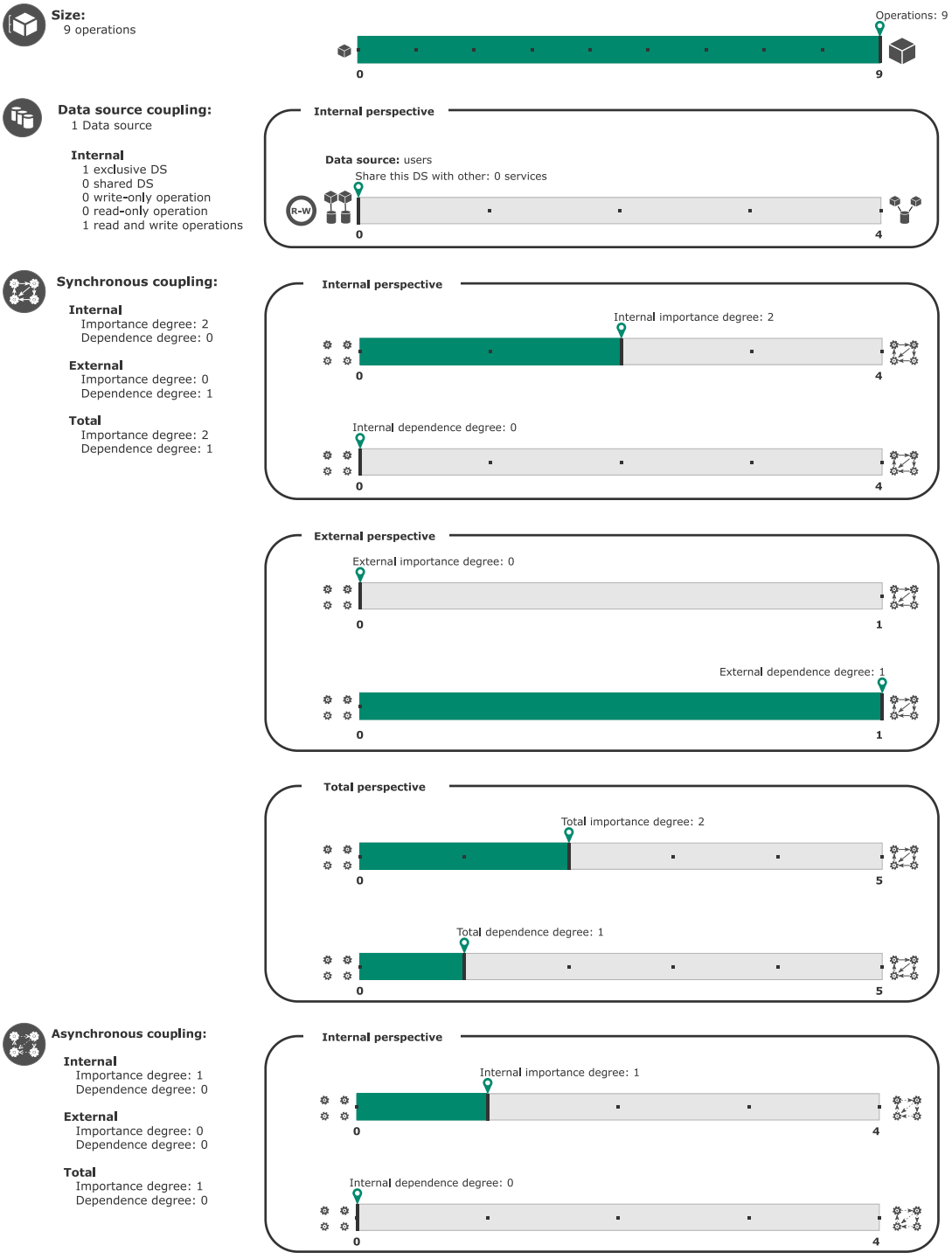


Fig. 21. User Service characterization.

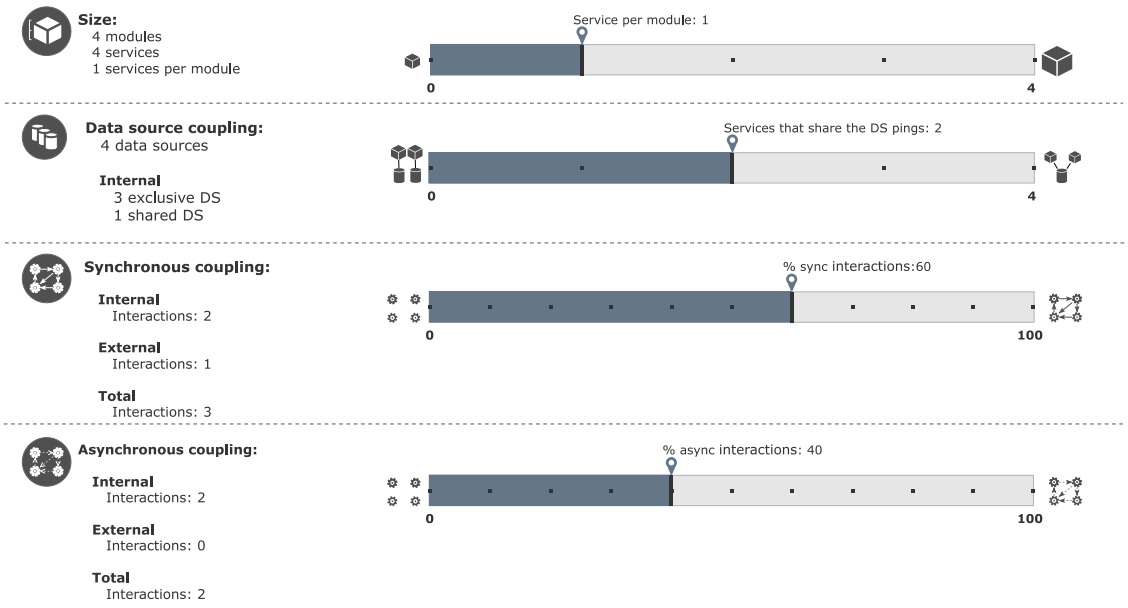


Fig. 22. Pingr characterization.

References

Alshuqayran, N., 2020. Static Microservice Architecture Recovery Using Model-driven Engineering (Ph.D. thesis). University of Brighton.

Alshuqayran, N., Ali, N., Evans, R., 2018. Towards micro service architecture recovery: An empirical study. In: 2018 IEEE International Conference on Software Architecture. ICSA, pp. 47–4709. http://dx.doi.org/10.1109/ICSA.2018.00014.

Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D., 2021. From monolithic systems to microservices: An assessment framework. Inf. Softw. Technol. 137, 106600. http://dx.doi.org/10.1016/j.infsof.2021.106600.

Baltes, S., Ralph, P., 2020. Sampling in software engineering research: A critical review and guidelines. Empir. Softw. Eng. 27, URL: https://arxiv.org/abs/2002.07764.

Baresi, L., Garriga, M., 2020. Microservices: The evolution and extinction of web services? In: Microservices: Science and Engineering. Springer International Publishing, pp. 3–28. http://dx.doi.org/10.1007/978-3-030-31646-4_1.

Basili, V.R., Rombach, H.D., 1988. The TAME project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. 14 (6), 758–773. http://dx.doi.org/10.1109/32.6156.

Bass, L., Clements, P., Kazman, R., 2012. Software Architecture in Practice, third ed. Addison-Wesley Professional, p. 1169.

Bianco, P., Kotermanski, R., Merson, P., 2007. Evaluating a Service-Oriented Architecture. Technical Report, Carnegie Mellon University.

Bogner, J., Schlinger, S., Wagner, S., Zimmermann, A., 2019. A Modular Approach to Calculate Service-Based Maintainability Metrics from Runtime Data of Microservices. In: Product-Focused Software Process Improvement. pp. 489–496. http://dx.doi.org/10.1007/978-3-030-35333-9_34.

Bogner, J., Wagner, S., Zimmermann, A., 2017a. Automatically measuring the maintainability of service- and microservice-based systems. In: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement on - IWSM Mensura '17. pp. 107–115. http://dx.doi.org/10.1145/3143434.3143443.

Bogner, J., Wagner, S., Zimmermann, A., 2017b. Towards a practical maintainability quality model for service and microservice-based systems. In: 11th European Conference on Software Architecture. ECSA 2017, pp. 195–198. http://dx.doi.org/10.1145/3129790.3129816.

Bogner, J., Wagner, S., Zimmermann, A., 2020a. Collecting service-based maintainability metrics from RESTful API descriptions: Static analysis and threshold derivation. In: Software Architecture. pp. 215–227. http://dx.doi.org/10.1007/978-3-030-59155-7_16.

Bogner, J., Zimmermann, A., Wagner, S., 2020b. Towards an evolvability assurance method for service-based systems. In: Advances in Service-Oriented and Cloud Computing. pp. 131–139. http://dx.doi.org/10.1007/978-3-030-63161-1_10.

Bonér, J., 2016. Reactive Microservices Architecture: Design Principles for Distributed Systems. O'Reilly, p. 54.

Bushong, V., Abdelfattah, A.S., Maruf, A.A., Das, D., Lehman, A., Jaroszewski, E., Coffey, M., Cerny, T., Frajtak, K., Tisnovsky, P., Bures, M., 2021. On microservice analysis and architecture evolution: A systematic mapping study. Appl. Sci. 11 (17), http://dx.doi.org/10.3390/app11177856.

Callegaro, M., Manfreda, K.L., Vehovar, V., 2015. Web Survey Methodology. Sage.

Cambridge University Press, 2022. Meaning of characterization in English. URL: https://dictionary.cambridge.org/dictionary/english/characterization.

Cardarelli, M., Iovino, L., Francesco, P.D., Salle, A.D., Malavolta, I., Lago, P., 2019. An extensible data-driven approach for evaluating the quality of microservice architectures. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. pp. 1225–1234. http://dx.doi.org/10.1145/3297280.3297400.

Cerny, T., Donahoo, M.J., Pechanec, J., 2017. Disambiguation and comparison of SOA, microservices and self-contained systems. In: Proceedings of the International Conference on Research in Adaptive and Convergent Systems. pp. 228–235. http://dx.doi.org/10.1145/3129676.3129682.

Cerny, T., Donahoo, M.J., Trnka, M., 2018. Contextual understanding of microservice architecture: Current and future directions. SIGAPP Appl. Comput. Rev. 17 (4), 29–45. http://dx.doi.org/10.1145/3183628.3183631.

Davis, F.D., 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. 13 (3), 319–340. http://dx.doi.org/10.2307/249008.

Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L., 2017. Microservices: Yesterday, today, and tomorrow. In: Present and Ulterior Software Engineering. Springer International Publishing, pp. 195–216. http://dx.doi.org/10.1007/978-3-319-67425-4_12.

Dreyfus, H.L., Dreyfus, S.E., Athanasiou, T., 1986. Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer. The Free Press.

Engel, T., Langermeier, M., Bauer, B., Hofmann, A., 2018. Evaluation of microservice architectures: A metric and tool-based approach. In: Information Systems in the Big Data Era. pp. 74–89. http://dx.doi.org/10.1007/978-3-319-92901-9_8.

Engström, E., Storey, M.-A., Runeson, P., Höst, M., Baldassarre, M.T., 2020. How software engineering research aligns with design science: a review. Empir. Softw. Eng. 25, 2630–2660. http://dx.doi.org/10.1007/s10664-020-09818-7.

del Esposte, A.D.M., 2018. A Scalable Microservice-based Open Source Platform for Smart Cities. University of São Paulo.

Ford, N., 2018. The State of Microservices Maturity – Survey Results. O'Reilly Media.

Francesco, P.D., Malavolta, I., Lago, P., 2017. Research on architecting microservices: Trends, focus, and potential for industrial adoption. In: 2017 IEEE International Conference on Software Architecture. ICSA, pp. 21–30. http://dx.doi.org/10.1109/ICSA.2017.24.

Granchelli, G., Cardarelli, M., Francesco, P.D., Malavolta, I., Iovino, L., Salle, A.D., 2017a. MicroART: A software architecture recovery tool for maintaining microservice-based systems. In: IEEE International Conference on Software Architecture Workshops. ICSAW 2017, pp. 298–302. http://dx.doi.org/10.1109/ICSAW.2017.9.

Granchelli, G., Cardarelli, M., Francesco, P.D., Malavolta, I., Iovino, L., Salle, A.D., 2017b. Towards recovering the software architecture of microservice-based systems. In: IEEE International Conference on Software Architecture

Workshops. ICSAW 2017, pp. 46–53. http://dx.doi.org/10.1109/ICSAW.2017.48.
Hassan, S., Bahsoon, R., 2016. Microservices and their design trade-offs: A self-adaptive roadmap. In: 2016 IEEE International Conference on Services Computing. SCC 2016, pp. 813–818. http://dx.doi.org/10.1109/SCC.2016.113.
Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28 (1), 75–105. http://dx.doi.org/10.2307/25148625.
Highscalability, 2020. One team at uber is moving from microservices to macroservices. URL: highscalability.com/blog/2020/4/8/one-team-at-uber-is-moving-from-microservices-to-macros/ervic.html.
InfoQ, 2020. To microservices and back again. URL: www.infoq.com/news/2020/04/microservices-back-again/.
innoQ, 2015. Self-contained systems: Assembling software from independent systems. URL: scs-architecture.org/index.html.
Jaramillo, D., Nguyen, D.V., Smart, R., 2016. Leveraging microservices architecture by using docker technology. In: SoutheastCon 2016. pp. 1–5. http://dx.doi.org/10.1109/SECON.2016.7506647.
Krafzig, D., Banke, K., Slama, D., 2005. Enterprise SOA: Service-Oriented Architecture Best Practices. Pearson Education.
Lederer, A.L., Maupin, D.J., Sena, M.P., Zhuang, Y., 2000. The technology acceptance model and the world wide web. Decis. Support Syst. 29 (3), 269–282. http://dx.doi.org/10.1016/S0167-9236(00)00076-2.
Lewis, J., Fowler, M., 2014. Microservices. URL: https://martinfowler.com/articles/microservices.html.
Mahmood, Z., 2007. The promise and limitations of service oriented architecture. Int. J. Comput. 1 (3), 74–78.
Martin, R.C., 2018. Clean Architecture: A Craftsmans Guide To Software Structure and Design. Prentice Hall, p. 409.
Mayer, B., Weinreich, R., 2018. An approach to extract the architecture of microservice-based software systems. In: 2018 IEEE Symposium on Service-Oriented System Engineering. SOSE, pp. 21–30. http://dx.doi.org/10.1109/SOSE.2018.00012.
Mendonça, N.C., Box, C., Manolache, C., Ryan, L., 2021. The monolith strikes back: Why istio migrated from microservices to a monolithic architecture. IEEE Softw. 38 (5), 17–22. http://dx.doi.org/10.1109/MS.2021.3080335.
Nadareishvili, I., Mitra, R., McLarty, M., Amundsen, M., 2016. Microservice Architecture: Aligning Principles, Practices, and Culture. O'Reilly Media.
Natis, Y., Schulte, R., 2003. Introduction to Service-Oriented Architecture. Technical Report, Gartner Group.
Newman, S., 2015. Building Microservices: Designing Fine-Grained Systems, first ed. O'Reilly Media, p. 259.
Newman, S., 2021. Building Microservices: Designing Fine-Grained Systems, second ed. O'Reilly Media, p. 616.
Ntentos, E., Zdun, U., Plakidas, K., Geiger, S., 2021a. Evaluating and improving microservice architecture conformance to architectural design decisions. In: Service-Oriented Computing. pp. 188–203. http://dx.doi.org/10.1007/978-3-030-91431-8_12.
Ntentos, E., Zdun, U., Plakidas, K., Geiger, S., 2021b. Semi-automatic feedback for improving architecture conformance to microservice patterns and practices. In: 2021 IEEE 18th International Conference on Software Architecture. ICSA, pp. 36–46. http://dx.doi.org/10.1109/ICSA51549.2021.00012.
Oxford University Press, 2022. Definition of model noun from the oxford advanced learner's dictionar. URL: https://www.oxfordlearnersdictionaries.com/definition/english/model_1?q=model.
Papazoglou, M.P., 2003. Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003.. WISE 2003, pp. 3–12. http://dx.doi.org/10.1109/WISE.2003.1254461.
Parnas, L.D., Clements, P.C., Weiss, D.M., 1985. The modular structure of complex systems. IEEE Trans. Softw. Eng. SE-11 (3), 259–266. http://dx.doi.org/10.1109/TSE.1985.232209.
Perepletchikov, M., Ryan, C., Frampton, K., Tari, Z., 2007. Coupling metrics for predicting maintainability in service-oriented designs. In: Australian Software Engineering Conference. ASWEC'07, pp. 329–338.
Rademacher, F., Sachweh, S., Zundorf, A., 2017. Differences between model-driven development of service-oriented and microservice architecture. In: 2017 IEEE International Conference on Software Architecture Workshops. ICSAW, pp. 38–45. http://dx.doi.org/10.1109/ICSAW.2017.32.
Raj, V., Sadam, R., 2021. Performance and complexity comparison of service oriented architecture and microservices architecture. Int. J. Commun. Netw. Distribut. Syst. 27 (1), 100–117. http://dx.doi.org/10.1504/ijcnds.2021.116463.
Ralph, P., Arshad, A., Ghaleb, T., 2022. Empirical standards checklist generator: Engineering research (aka design science). URL: https://acmsigsoft.github.io/EmpiricalStandards/docs/?standard=EngineeringResearch.
Richards, M., 2015. Software Architecture Patterns. O'Reilly Media.
Richards, M., 2016. Microservices Vs. Service-Oriented Architecture. O'Reilly Media, p. 44.
Richardson, C., 2018. Microservices Patterns. Manning Publicatins Co., p. 489.
Richardson, C., Smith, F., 2016. Microservices from Design to Deployment. NGINX, p. 74.
Rosa, T.D.O., Daniel, J.F.L., Guerra, E.M., Goldman, A., 2020a. A method for architectural trade-off analysis based on patterns: Evaluating microservices structural attributes. In: Proceedings of the European Conference on Pattern Languages of Programs 2020. p. 8. http://dx.doi.org/10.1145/3424771.3424809.
Rosa, T.D.O., Goldman, A., Guerra, E.M., 2020b. How 'micro' are your services? In: IEEE International Conference on Software Architecture Companion. ICSA-C 2020, pp. 75–78.
Rosa, T., Goldman, A., Guerra, E., 2020c. Modelo para Caracterização e Evolução de Sistemas com Arquitetura Baseada em Serviços. In: Anais Estendidos do XI Congresso Brasileiro de Software: Teoria e Prática. pp. 38–46. http://dx.doi.org/10.5753/cbsoft_estendido.2020.14607.
Rosa, T.D.O., Guerra, E.M., Correia, F.F., Goldman, A., 2022. CharM: A Model for Characterization of Serviced-Based Systems Architecture. Zenodo, http://dx.doi.org/10.5281/zenodo.6590692.
Salah, T., Zemerly, M.J., Chan, Y.Y., Al-Qutayri, M., Al-Hammadi, Y., 2016. The evolution of distributed systems towards microservices architecture. In: 2016 11th International Conference for Internet Technology and Secured Transactions. ICITST, pp. 318–325. http://dx.doi.org/10.1109/ICITST.2016.7856721.
Santana, E., Rosa, T., Daniel, J., Goldman, A., 2021. Desenvolvendo o Sorting Hat: uma Ferramenta para Caracterização de Arquitetura Baseada em Serviços. In: Anais Estendidos do XII Congresso Brasileiro de Software: Teoria e Prática. pp. 127–136. http://dx.doi.org/10.5753/cbsoft_estendido.2021.17298.
Shadija, D., Rezai, M., Hill, R., 2017. Towards an understanding of microservices. In: 2017 23rd International Conference on Automation and Computing. ICAC, pp. 1–6. http://dx.doi.org/10.23919/IConAC.2017.8082018.
Shim, B., Choue, S., Kim, S., Park, S., 2008. A design quality model for service-oriented architecture. In: 15th Asia-Pacific Software Engineering Conference. pp. 403–410. http://dx.doi.org/10.1109/APSEC.2008.32.
Sneed, H.M., 2006. Integrating legacy software into a service oriented architecture. In: Conference on Software Maintenance and Reengineering. CSMR'06, pp. 11 pp.–14. http://dx.doi.org/10.1109/CSMR.2006.28.
Soldani, J., Tamburri, D.A., Heuvel, W.-J.V.D., 2018. The pains and gains of microservices: A systematic grey literature review. J. Syst. Softw. 146, 215–232. http://dx.doi.org/10.1016/j.jss.2018.09.082.
Statista, 2022. Software developer gender distribution worldwide as of 2021. URL: https://www.statista.com/statistics/1126823/worldwide-developer-gender/.
Thönes, J., 2015. Microservices. IEEE Softw. 32 (1), 116. http://dx.doi.org/10.1109/MS.2015.11.
Vera-Rivera, F.H., Gaona, C., Astudillo, H., 2021. Defining and measuring microservice granularity—a literature overview. PeerJ Computer Science 7, http://dx.doi.org/10.7717/peerj-cs.695.
Wagner, S., Mendez, D., Felderer, M., Graziotin, D., Kalinowski, M., 2020. Challenges in survey research. In: Contemporary Empirical Methods in Software Engineering. Springer International Publishing, pp. 93–125. http://dx.doi.org/10.1007/978-3-030-32489-6_4.
Wolff, E., 2016a. Self-contained systems: A different approach to microservices. URL: https://www.innoq.com/en/articles/2016/11/self-contained-systems-different-microservices/#self-containedsystems.
Wolff, E., 2016b. Services: SOA, microservices und self-contained systems. URL: https://www.innoq.com/en/articles/2016/11/services-soa-microservices-scs/#fazit.
Zdun, U., Navarro, E., Leymann, F., 2017. Ensuring and assessing architecture conformance to microservice decomposition patterns. In: Service-Oriented Computing. pp. 411–429. http://dx.doi.org/10.1007/978-3-319-69035-3_29.
Zimmermann, O., 2017. Microservices tenets: Agile approach to service development and deployment. Comput. Sci. Res. Dev. 32 (3), 301–310. http://dx.doi.org/10.1007/s00450-016-0337-0.

Thatiane de Oliveira Rosa is currently an assistant professor at the Federal Institute of Tocantins (IFTO), Brazil. She received her Ph.D. (2023) degree in Computer Science from the University of Sao Paulo (USP). Her research focus is Software Engineering, emphasizing Software Architecture, Good Practices for Agile Development, and Teaching Methods in Computer Science.

Eduardo Martins Guerra is currently a researcher at the Free University of Bozen-Bolzano, in Italy. His research interests include agile methods, software patterns, framework development, software analytics, and dynamic architectures. Guerra received a Ph.D. in computer engineering from the Aeronautics Institute of Technology and has practical experience in architecture and framework design.

Filipe Figueiredo Correia is an assistant professor at the Faculty of Engineering of the University of Porto (FEUP), in Portugal. His research focus is in Software Engineering topics, namely on Software Architecture, Design Patterns, Continuous Delivery, Agile Methods and Live Software Development. Filipe is part of the Software Engineering Group at FEUP and of INESC TEC.

Alfredo Goldman holds a PhD from INPG, Grenoble. Currently, he is an associate professor at USP. Subject area editor of Parallel Computing. He was co-program chair in 2014 of IEEE-NCA and SBAC-PAD, in 2015 of WSCAD, and in 2022 of SEMISH. He was the track chair of EuroPar 2017 and 2021, IEEE SCC 2020, and CARLA 2022. He was co-general chair in IEEE-NCA 2017 and SBAC-PAD 2018. His main research interests are parallel and distributed computing, scheduling, and agile methods. He is on the board of governors of the Brazilian Computer Society and a member of ACM and IEEE-CS.