Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Angelo Joppert

InA^{2}rMS: Instrumento de apoio à Avaliação da Arquitetura de MicrosServiço

Rio de Janeiro
2023

Angelo Joppert

InA^{2}rMS: Instrumento de apoio à Avaliação da Arquitetura de MicrosServiço

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Marcelo Schots de Oliveira

Rio de Janeiro

2023

CATALOGAÇÃO NA FONTE

UERJ/REDE SIRIUS/BIBLIOTECA CTC/A

J81 Joppert, Angelo.
InA²rMS: instrumento de apoio à avaliação da arquitetura de microsserviço/ Angelo Joppert. – 2023.
170 f.: il.

Orientador: Marcelo Schots de Oliveira
Dissertação (Mestrado em Ciências Computacionais) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Arquitetura de software - Teses. 2. Software - Desenvolvimento - Teses. I. Oliveira, Marcelo Schots de. II. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. III. Título.

CDU 004.41

Patricia Bello Meijinhos CRB7/5217 - Bibliotecária responsável pela elaboração da ficha catalográfica

_____ _____

Assinatura Data

Angelo Joppert

InA^{2}rMS: Instrumento de apoio à Avaliação da Arquitetura de MicrosServiço

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Ciências Computacionais, da Universidade do Estado do Rio de Janeiro.

Aprovada em 02 de agosto de 2023.
Banca Examinadora:

Prof. Dr. Marcelo Schots de Oliveira (Orientador)
Instituto de Matemática e Estatística - UERJ

Prof. Dr. Wesley Klewerton Guez Assunção
Johannes Kepler University Linz

Prof.^{a} Dr.^{a} Flávia Maria Santoro
Instituto de Matemática e Estatística - UERJ

Rio de Janeiro
2023

DEDICATÓRIA

A meus pais Ney e Dora, a minha esposa Luana e a nossa filha Jade.

AGRADECIMENTOS

Primeiramente, gostaria de expressar minha gratidão a Deus por conceder-me a vida, por instruir meu caminho e por ter me dado forças para a realização deste trabalho.

À minha esposa, Luana, uma verdadeira amiga e incentivadora, agradeço pela compreensão da minha ausência e por todo o apoio despendido nesta jornada. Seu amor incondicional e suporte foram fundamentais para o meu sucesso.

À minha família e amigos, gostaria de agradecer pelo amor, apoio e compreensão durante todo o processo. O incentivo constante e as palavras de encorajamento que recebi de vocês foram essenciais para me manter motivado e focado em alcançar meus objetivos.

Ao meu orientador Marcelo Schots de Oliveira, pelo seu profissionalismo, por seu constante apoio, orientação e paciência ao longo deste trabalho.

Gostaria de estender meu sincero agradecimento aos especialistas que participaram dessa pesquisa, pois sem o compartilhamento de seus conhecimentos e experiências esse trabalho não seria possível. A contribuição de vocês foi inestimável.

Aos membros da banca examinadora pela disponibilidade e aceite em avaliar este trabalho.

Aos professores e amigos do Programa de Pós-Graduação em Ciências Computacionais da Universidade do Estado do Rio de Janeiro por todos os conhecimentos transmitidos, que foram essenciais para aprimorar minha formação acadêmica e profissional.

Agradeço à Marinha do Brasil pela oportunidade de me especializar em minha área de formação com essa importante capacitação.

Por fim, agradeço a todos que contribuíram para o sucesso dessa dissertação.

O que não é medido não é gerenciado.

Robert Kaplan e David Norton

RESUMO

JOPPERT, Angelo. *InA²rMS: Instrumento de apoio à Avaliação da Arquitetura de MicrosServiço*. 2023. 170 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

A demanda por abordagens mais eficientes e sistematizadas de desenvolvimento de sistemas é cada vez maior. Alguns estilos arquiteturais surgiram como respostas às abordagens tradicionais monolíticas para a construção de sistemas. Por isso, grandes empresas têm migrado da arquitetura monolítica (AMO) para a arquitetura de microsserviços (AMS). A AMS ficou em evidência na última década, devido à sua adequabilidade às tecnologias nativas de nuvem e à sua natureza distribuída. A AMS pode prover diversos benefícios para as organizações, como a melhora no gerenciamento e reutilização de funcionalidades, na agilidade, na manutenção, no processo de liberação e de implantação, aprimorando a eficiência dos custos relacionados com o ciclo de vida do *software*. Devido à sua complexidade, a AMS torna-se desafiadora para arquitetos e desenvolvedores de *software*. Considerando sua natureza distribuída, seu desenvolvimento requer uma compreensão clara das características necessárias e dos recursos envolvidos para implementá-las. Diante desse cenário, a análise conceitual das características em relação ao domínio, com a ponderação de sua pertinência e com a identificação de seus relacionamentos, pode subsidiar a avaliação do atendimento às características desta arquitetura. Neste sentido, este trabalho visa fornecer uma ferramenta para apoiar à análise e avaliação de microsserviços (MS) em relação ao atendimento às características da AMS, de forma que seja possível expressar o grau de atendimento dessas características por meio de uma estratégia de visualização de informação. Para o atingimento desse objetivo, propõe-se uma abordagem que consiste de um modelo de características, um glossário de termos, um questionário de avaliação integrado a uma técnica de visualização de dados, para apoiar arquitetos e desenvolvedores durante a avaliação de MS. Foram conduzidos estudos com o propósito de verificar a adequabilidade e aplicabilidade da abordagem. Os resultados fornecem evidências positivas quanto à adoção da abordagem em processos de construção e de manutenção de MS. Acredita-se que o presente trabalho pode auxiliar, de forma holística, arquitetos e equipes de desenvolvimento no entendimento e na avaliação das diversas características envolvidas na AMS, aprimorando o processo de desenvolvimento como um todo.

Palavras-chave: Arquitetura de microsserviços. Modelo de características. Interesses não-funcionais.

ABSTRACT

JOPPERT, Angelo. *InA^{2}rMS: Microservice architecture evaluation support instrument.* 2023. 170 f. Dissertação (Mestrado em Ciências Computacionais) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

The demand for more efficient and systematic approaches to system development is constantly increasing. Some architectural styles have emerged as responses to traditional monolithic approaches to system construction. As a result, large companies have been transitioning from Monolithic Architecture (MOA) to Microservices Architecture (MSA). MSA has gained prominence in the last decade due to its suitability for cloud-native technologies and its distributed nature. The MSA can provide several benefits for organizations, such as improvement in feature management and reuse, agility, maintenance, deployment, and delivery processes, enhancing cost efficiency related to the software lifecycle. Due to its complexity, MSA becomes challenging for architects and software developers. Considering its distributed nature, its development requires a clear understanding of the necessary features and the resources involved to implement them. In light of this scenario, the conceptual analysis of the features in relation to the domain, weighing their relevance and identifying their relationships, can support the evaluation of the compliance to this architecture's features. In this sense, this work aims to provide a tool to support the analysis and evaluation of microservices (MS) concerning their adherence to the features of MSA, allowing the degree of compliance with these features to be expressed through an information visualization strategy. To achieve this objective, an approach is proposed that consists of a features model, a glossary of terms, an evaluation questionnaire integrated with a data visualization technique, to support architects and developers during the evaluation of MS. Studies were conducted with the purpose of verifying the suitability and applicability of the approach. The results provide positive evidence regarding the adoption of the approach in the processes of building and maintaining MS. It is believed that this work can holistically assist architects and development teams in understanding and evaluating the diverse features involved in MSA, enhancing the overall development process.

Keywords: Microservices architecture. Feature model. Non-functional concerns.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

Motivação

Um mundo volátil, incerto e complexo é consequência da transformação digital trazida pela quarta revolução industrial, também conhecida como indústria 4.0. Essa revolução tecnológica transformou a maneira como as organizações se relacionam com seus funcionários, parceiros e clientes, tornando o *software* uma parte essencial dos negócios em todos os domínios (SCHWAB, 2016).

As organizações enfrentam a necessidade de criar, de forma ágil, sistemas escaláveis que possuam impacto em novas formas de produção e organização empresarial. A tradicional arquitetura monolítica (AMO) não atende mais às necessidades de escalabilidade e de desenvolvimento rápido (LEON et al., 2020). A todo momento, identifica-se a demanda por abordagens mais eficientes e sistematizadas de desenvolvimento de sistemas, que permitam o aprimoramento dos mecanismos de controle, tendo como meta o aumento da produtividade (MARZULLO, 2014).

De acordo com Richards (2016), alguns estilos arquiteturais baseados em serviços, que demandam por uma arquitetura modular e que encapsulam funcionalidades em serviços, surgiram como respostas às abordagens tradicionais monolíticas para a construção de sistemas. Nesse cenário, segundo Kazanavičius e Mažeika (2019), a arquitetura de microsserviços (AMS) ficou em evidência na última década, devido à sua adequabilidade às tecnologias nativas de nuvem (VALE et al., 2022) e à sua natureza distribuída.

Além disso, há uma crescente demanda da transição da AMO tradicional, com baixa reusabilidade e com alto custo de manutenção, para uma AMS, mais dinâmica e flexível. Tal transição, além de estar em conformidade com o cenário tecnológico atual, envolve o uso de automação com a adoção de diversas práticas de engenharia de *software* (CARVALHO et al., 2019).

Tal arquitetura pode prover benefícios para as organizações, como a melhora no gerenciamento e reutilização de funcionalidades, na agilidade, na manutenção, no processo de liberação e de implantação, aprimorando dessa maneira, a eficiência dos custos relacionados com o ciclo de vida de sistemas (WOLFART et al., 2021). Devido a esses benefícios, grandes empresas como Walmart, Spotify, Netflix, Amazon e eBay migraram da AMO para a AMS (LEON et al., 2020).

No entanto, segundo Araujo (2019), as características descritas pela AMS tornam esse padrão arquitetural muito heterogêneo. Devido à sua complexidade, por se tratar de uma arquitetura distribuída, sua implementação não é trivial, exigindo não só uma visão abrangente de todo o ecossistema, como também diretrizes e orientações que possam nortear organizações na sua adoção ou validação. Por essas razões, a implementação de

AMS pode se tornar desafiadora para arquitetos de *software*.

Além disso, a notável relação de interdependência entre as diferentes características da AMS requer uma tomada de decisão cuidadosa, de modo a priorizar algumas características mais relevantes ao domínio em detrimento de outras (*e.g.*, segurança *vs.* desempenho em microsserviços relacionados ao domínio bancário). A desconsideração de interesses não-funcionais orientados ao domínio ou mesmo a ausência de formalização dessas decisões arquiteturais pode resultar em problemas em curto e longo prazo.

Segundo Kang et al. (1990), o desenvolvimento de sistemas de *software* complexos requer uma compreensão das características desejadas e dos recursos necessários. Assim, a modelagem das principais características no contexto de MS, com a captura de semelhanças e variabilidades, definindo suas relações estruturais e de suas interdependências, pode subsidiar um instrumento de avaliação do atendimento a esses interesses.

Nesse contexto, pretende-se, com este trabalho, fornecer um instrumento de apoio à avaliação das características da AMS, permitindo expressar o grau de atendimento às características por meio de uma estratégia de visualização de informação. Tal instrumento pode auxiliar organizações no entendimento e avaliação de seus microsserviços em relação às características relevantes ao cenário, contribuindo para o aprimoramento de produtos e serviços, e melhorando a eficiência dos custos relacionados com o ciclo de vida de sistemas.

Aplicabilidade

Na Administração Pública, organizações que possuem unidades fisicamente distantes e que, durante décadas, desenvolveram seus sistemas de forma descentralizada, são caracterizadas por apresentar tecnologias não padronizadas e um considerável número de sistemas com funcionalidades e dados redundantes, dificultando a modernização e a integração desses sistemas.

Esses sistemas são considerados legados por possuírem uma vida útil longa, com tecnologia obsoleta e arquitetura degradada, exigindo uma grande quantidade de recursos para manutenção e dificultando a transformação digital e a inovação.

Atualmente, observa-se uma tendência na adoção de AMS como estratégia para a modernização de sistemas legados monolíticos, tratando-se de um processo longo e desafiador, que demanda a análise de aspectos técnicos, operacionais e organizacionais, a fim de evitar fracassos e desperdício de recursos (WOLFART et al., 2021).

Além disso, diante de um cenário com recursos limitados e orçamentos reduzidos, a inovação tecnológica na Administração Pública é uma das soluções para a redução de custos. Governos em todo o mundo enfrentam desafios constantes para promover um número cada vez maior de serviços públicos com volume de gastos cada vez menor a cada ano (SILVA, 2020).

Assim, os benefícios relacionados com a utilização da AMS a longo prazo, além da redução de custos com o ciclo de desenvolvimento e de manutenção de *software*, podem ajudar organizações a responderem de forma mais rápida às suas demandas, melhorando a eficiência e a racionalização do uso de recursos materiais e humanos.

Problema

O problema tratado nessa dissertação pode ser sintetizado da seguinte forma: *Como* ***identificar***, ***expressar*** *e* ***avaliar*** *o grau de atendimento de um MS em relação aos interesses não-funcionais (concerns) relevantes da AMS?*

Questões de pesquisa

Este trabalho tem como objetivo responder às seguintes questões de pesquisa:

QP1. Quais características são relevantes da AMS?

QP2. Como as características relevantes da AMS se relacionam?

QP3. Como analisar e avaliar microsserviços em relação às características da AMS?

QP4. Como expressar o grau de atendimento de microsserviços às características de uma forma intuitiva e precisa?

Objetivos

Os objetivos desta dissertação são:

- Analisar o conceito de AMS, identificando características relevantes com base em uma análise da literatura;
- Propor um modelo de características sobre a AMS de acordo com a análise conceitual do item anterior;
- Relacionar as características interdependentes com base no modelo de características proposto;
- Definir as características identificadas por meio de um glossário de termos;

- Propor uma abordagem para análise e avaliação de MS em relação aos interesses não-funcionais, presentes no modelo de características proposto, por meio de uma estratégia de visualização de dados para expressar o atendimento às características.

Organização do texto

Essa introdução dedicou-se à apresentação da motivação, da aplicabilidade, do problema, das questões de pesquisa e dos objetivos desse projeto de pesquisa. Os capítulos do presente trabalho estão organizados da seguinte forma:

O Capítulo 1 apresenta conceitos básicos sobre a arquitetura de *software*. Além disso, as arquiteturas monolítica, orientada a serviços e de microsserviços são detalhadas e comparadas. Por fim, são apresentados conceitos sobre modelagem de características, características não-funcionais em arquitetura de *software* e também sobre visualização de dados.

O Capítulo 2 apresenta os trabalhos relacionados ao problema tratado nessa dissertação. Os trabalhos são apresentados em duas categorias: (i) estudos sobre métodos e técnicas de avaliação arquitetural e (ii) estudos sobre classificações e taxonomias de AMS.

O Capítulo 3 apresenta a metodologia e as atividades necessárias para se atingir os objetivos deste trabalho. Também são apresentados os critérios e a estratégia utilizados.

O Capítulo 4 apresenta a abordagem desenvolvida para a análise e avaliação de MS em relação aos interesses não-funcionais.

O Capítulo 5 descreve o experimento e suas etapas. O planejamento apresenta o detalhamento dos artefatos sendo avaliados, os perfis dos participantes, os cenários de avaliação, os tamanho de amostras e as métricas sendo avaliadas, A execução detalha as amostras e os dados coletados, enquanto a discussão dos resultados analisa os resultados obtidos.

O Capítulo intitulado “Considerações finais” sumariza os pontos centrais do presente estudo, suas contribuições e limitações, além de apresentar sugestões para trabalhos futuros.

Os Apêndices A, B, C, apresentam, respectivamente, o modelo de características, o glossário de termos e o questionário de avaliação.

Os Apêndices D, E, F, G, apresentam, respectivamente, os dados coletados na avaliação: do modelo de características, do glossário de termos, do questionário de avaliação e da técnica de visualização.

Por fim, o Anexo apresenta o FMCheck (*Feature Model Checklist*), uma técnica de inspeção baseada em *checklist* com o objetivo de apoiar a detecção de defeitos em *feature models*. Essa técnica é utilizada neste estudo para avaliar o modelo de características proposto.

1 REFERENCIAL TEÓRICO

Neste capítulo, são apresentados os conceitos e definições sobre arquitetura de *software*. Além disso, as arquiteturas monolítica, orientada a serviços e de microsserviços são apresentadas e comparadas. Por fim, são apresentados conceitos sobre a modelagem de características e também sobre características não-funcionais em arquitetura de *software*.

1.1 Arquitetura de software

Segundo Pressman (2009), a arquitetura é a estrutura ou a organização de componentes de programa, a maneira através da qual esses componentes interagem e a estrutura de dados utilizada pelos componentes. De acordo com Hansen e Sun (2011), a arquitetura de um sistema pode ser definida como um arranjo dos elementos funcionais relativamente abstratos de um sistema em vários blocos de construção físicos, com o mapeamento dos elementos funcionais para componentes físicos e a especificação das interfaces entre os componentes físicos que interagem.

A arquitetura de *software* surgiu como um conceito para o desenvolvimento bem-sucedido de sistemas grandes e complexos (CLEMENTS et al., 2003), sendo considerada a estrutura fundamental de um *software*, definindo os requisitos técnicos e operacionais, e tornando-se responsável pela otimização dos atributos de uma aplicação, como eficiência, gerenciabilidade, escalabilidade, confiabilidade, modificabilidade, implantabilidade entre outros aspectos (GOS; ZABIEROWSKI, 2020).

Desde o final da década de 1980, essa área tem amadurecido abrangendo um amplo conjunto de notações, ferramentas e técnicas de análise, fornecendo orientação concreta para projetos de desenvolvimento de *software* (SHAW; CLEMENTS, 2006).

De acordo com Clements et al. (2010), a arquitetura de *software* tem emergido como uma sub-disciplina importante da engenharia de *software*, tornando-se uma base conceitual amplamente aceita para o desenvolvimento de *software* por organizações de todos os tamanhos, em todas as áreas de aplicação (STAFFORD; CLEMENTS, 2007).

1.2 Da arquitetura monolítica à arquitetura de microsserviços

A evolução dos sistemas computacionais foi impulsionada pelo menor tamanho, pelo menor custo e pelo melhor desempenho do *hardware*. São décadas de evolução onde a arquitetura de *software* teve um papel fundamental na definição de diversos padrões de construção de *software*.

Segundo Cojocaru, Oprescu e Uta (2019), recentemente a arquitetura de microsserviços (AMS) vem ganhando força com seus princípios baseados nos da arquitetura orientada a serviços (AOS) (do inglês *Service Oriented Architecture* - SOA), distanciando-se cada vez mais da arquitetura monolítica (AMO).

No escopo desse trabalho, o presente capítulo abordará a arquitetura monolítica (AMO), a arquitetura orientada a serviços (AOS) e a arquitetura de microsserviços (AMS).

Uma forma de compreensão dessas arquiteturas é iniciar pela análise da granularidade de suas funcionalidades ou de seus serviços, incluindo como essas podem impactar na implantação das versões referentes a cada arquitetura. A Figura 1 apresenta as arquiteturas supracitadas e suas respectivas granularidades.

Figura 1 - Granularidades das arquiteturas: AMO, AOS e AMS.

Fonte: Adaptado de Samarpit (2018).

- AMO: caracterizada como uma unidade única (*single unit*), ou "grão único", contendo todas as funcionalidades do sistema, havendo a necessidade do empacotamento dos componentes de *software* da aplicação, formando uma única unidade de implantação;
- AOS: caracterizada como uma granularidade grossa (*coarse-grained*), ou pela existência de poucos "grãos" ou serviços, em que cada um contém um conjunto de operações, que pode não ser o mínimo necessário. Essa granularidade possui uma melhora na implantabilidade, porém não permite implantações totalmente isoladas devido a um nível de acoplamento razoável entre os serviços, caracterizado pela amplitude do contexto funcional dos serviços dessa arquitetura; e
- AMS: caracterizada como uma granularidade fina (*fine-grained*), ou pela existência de muitos "grãos" ou serviços, de tamanho pequeno, em que cada um contem um conjunto mínimo de operações. Essa granularidade é a que mais favorece a implantabilidade, pois permite implantações isoladas de outros serviços.

1.2.1 Arquitetura monolítica (AMO)

Segundo Kazanavičius e Mažeika (2019), a AMO é a forma tradicional de desenvolvimento de *software* na qual todas as funções são encapsuladas em uma única aplicação. O *software* monolítico é projetado para ser autocontido. Este tipo de arquitetura é fortemente acoplada, significando que se um dos componentes não estiver presente, o sistema não será executado ou compilado.

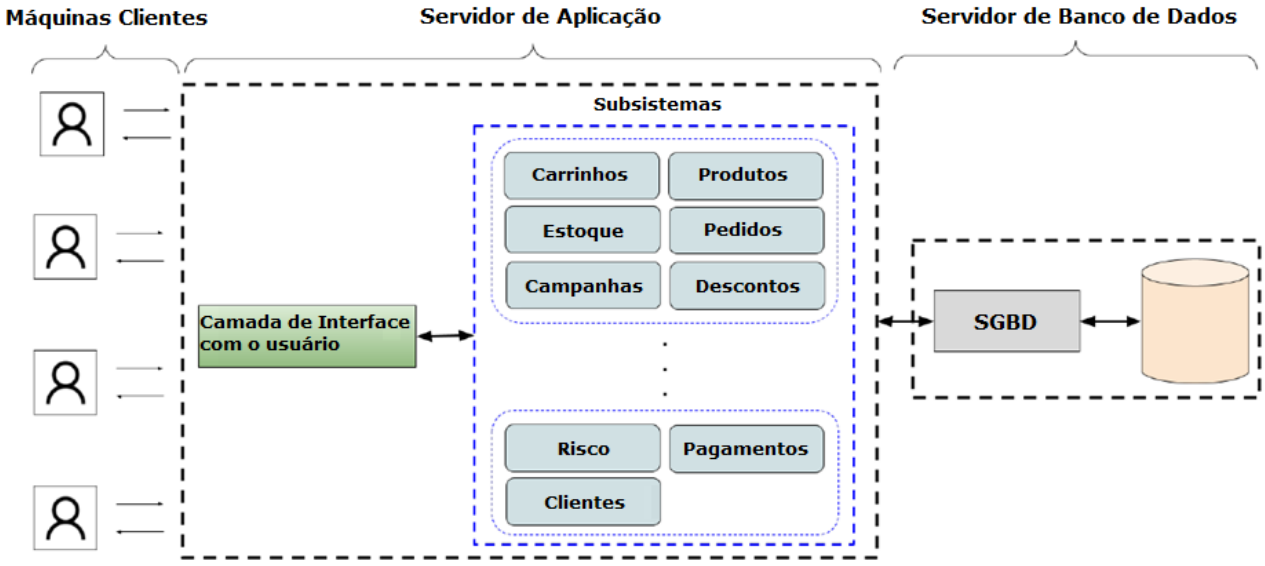
De acordo com Lehtola (2020), um sistema monolítico precisa ser desenvolvido e implantado como uma unidade única, criando desafios à medida que o sistema cresce em tamanho, pois a adição de novos recursos se torna mais difícil e alterações podem gerar efeitos desconhecidos.

Conforme o tamanho do código aumenta, as interdependências também crescem, gerando um efeito em cascata em todo o sistema, na ocorrência de falhas. Além disso, restrições de tecnologia e questões de escalabilidade são um problema para a entrega contínua, afetando a rapidez em se lidar com necessidades emergentes do negócio (MUNAF et al., 2019).

Outro aspecto a ser observado é o desempenho, que é afetado quando a quantidade de dados a ser processada aumenta ou excede um certo nível de capacidade (TAPIA et al., 2020).

A Figura 2 apresenta a arquitetura monolítica tradicional, onde o monólito, representado pelo servidor de aplicação (*Application Server Machine*), contem módulos e subsistemas totalmente integrados, que cooperam de forma centralizada.

Figura 2 - Arquitetura monolítica tradicional.



Fonte: Adaptado de Laigner et al. (2021).

As seguintes vantagens da AMO podem ser mencionadas (KAZANAVIČIUS; MAŽEIKA, 2019):

- **Implantação simplificada:** apenas uma aplicação precisa ser implantada;
- **Menos preocupações transversais**: algumas preocupações tendem a ser transversais (*i.e.*, desempenho, segurança, monitoramento, registro de *logs* etc.), afetando as funcionalidades principais em um sistema. Como o monólito contém subsistemas e módulos integrados, torna-se mais simples integrar o código aos componentes relacionados às preocupações transversais, já que todos estão rodando no mesmo contexto em um sistema; e
- **Sobrecarga operacional reduzida:** apenas uma aplicação precisa ser configurada.

As desvantagens da AMO são as seguintes:

- **Alto acoplamento**: se um dos componentes não estiver presente, o sistema não é compilado ou executado (KAZANAVIČIUS; MAŽEIKA, 2019);
- **Confiabilidade limitada**: um defeito em qualquer componente pode potencialmente comprometer todo o sistema (MUNAF et al., 2019); e
- **Degradação do desempenho**: o desempenho pode ser afetado quando a quantidade de dados a ser processada excede um certo nível de capacidade (TAPIA et al., 2020).

Normalmente, sistemas legados estão sempre crescendo em tamanho e complexidade. Com o tempo, as desvantagens da AMO podem superar as vantagens. Ao chegar nesse ponto, as organizações começam a procurar novas soluções arquiteturais.

1.2.2 Arquitetura orientada a serviços (AOS)

Segundo Newman (2015), AOS é uma abordagem de projeto arquitetural em que vários serviços colaboram entre si para fornecer um conjunto de funcionalidades. A AOS surgiu como uma abordagem para combater os desafios das AMO, sendo baseada em oito princípios de projeto da orientação a serviços, como ilustra a Figura 3.

Figura 3 - Os oito princípios de projeto(*design*) da orientação a serviços.

Fonte: Adaptado de Erl (2009).

AOS é uma das arquitetura de *software* mais populares, amplamente adotada em sistemas distribuídos. AOS trouxe a ideia de serviços interoperáveis, fracamente acoplados, reutilizáveis, componíveis e autônomos. Esse padrão utiliza *interfaces* de serviços com uma linguagem de comunicação comum em uma rede. O *enterprise service bus* (ESB), um dos componentes mais importantes, é um barramento de serviços corporativos que disponibiliza os serviços do sistema para os usuários e outras aplicações, acelerando os processos de integração. (FAHMI; HUANG; WANG, 2020).

A essência da proposta da AOS é muito sensata, no entanto, apesar de vários esforços, existe uma falta de consenso em sua adoção. Além disso, alguns problemas atribuídos à AOS são, na verdade, problemas com protocolos de comunicação (*e.g*, SOAP - *Simple Object Access Protocol*), *middleware* de fornecedores, falta de orientação sobre a granularidade de serviços ou sobre como dividir sistemas (NEWMAN, 2015).

Ademais, os serviços na AOS são frequentemente implementados como aplicativos monolíticos e implantados como uma unidade única. Consequentemente, a escalabilidade e a manutenibilidade do serviço são impactadas, pois não podem ser realizadas isoladamente. Assim, a mudança de um único serviço pode afetar inteiramente o sistema. (WANG; FAHMI, 2018).

Conforme Kistasamy, Merwe e Harpe (2010), as seguintes propriedades devem ser identificadas e ser predominantes em uma AOS:

- **Fracamente acoplada**: a arquitetura é composta por vários serviços conectados de forma que sejam resilientes a falhas e a latência de uma rede;
- **Componível**: aplicações utilizando AOS são criadas pela composição de serviços bem testados;

- **Detectável e dinâmica**: os serviços são descobertos por meio de um diretório e vinculados em tempo de execução, ao invés de tempo de compilação;
- **Interoperável**: padrões garantem que os serviços de diferentes organizações possam ser utilizados entre si;
- **Localmente transparente**: os serviços são construídos de tal forma que o sistema desconhece a localização dos vários serviços; e
- **Propriedades compartilhadas**: como um serviço pode ser composto por serviços que pertençam a entidades externas, a *interface* de serviço publicada será tratada como uma caixa-preta do ponto de vista de programadores, uma vez que eles não podem modificar o código.

De acordo com Dirksen (2012), a AOS possui as seguintes vantagens:

- **Agilidade** no lançamento de novos produtos e serviços em resposta às mudanças do mercado;
- **Melhor interoperabilidade**: com a utilização de contratos bem definidos, baseados em padrões;
- **Melhora na reutilização**, com o apoio de um inventário de serviços;
- **Melhora na qualidade de *software***, com o apoio de um conjunto de padrões definidos e das melhores práticas para a criação de serviços; e
- **Redução de custos**, com a reutilização baseada em padrões desenvolvimento.

No entanto, de acordo com (ABRAMS; SCHULTE, 2008), algumas organizações consideram baixo o nível de reusabilidade alcançado. Além disso, empresas que utilizam a AOS enfrentam desafios nas áreas de governança, teste, configuração, controle de versão, gerenciamento de metadados, monitoramento de nível de serviço, segurança e interoperabilidade, dentre outras.

1.2.3 Arquitetura de microsserviços (AMS)

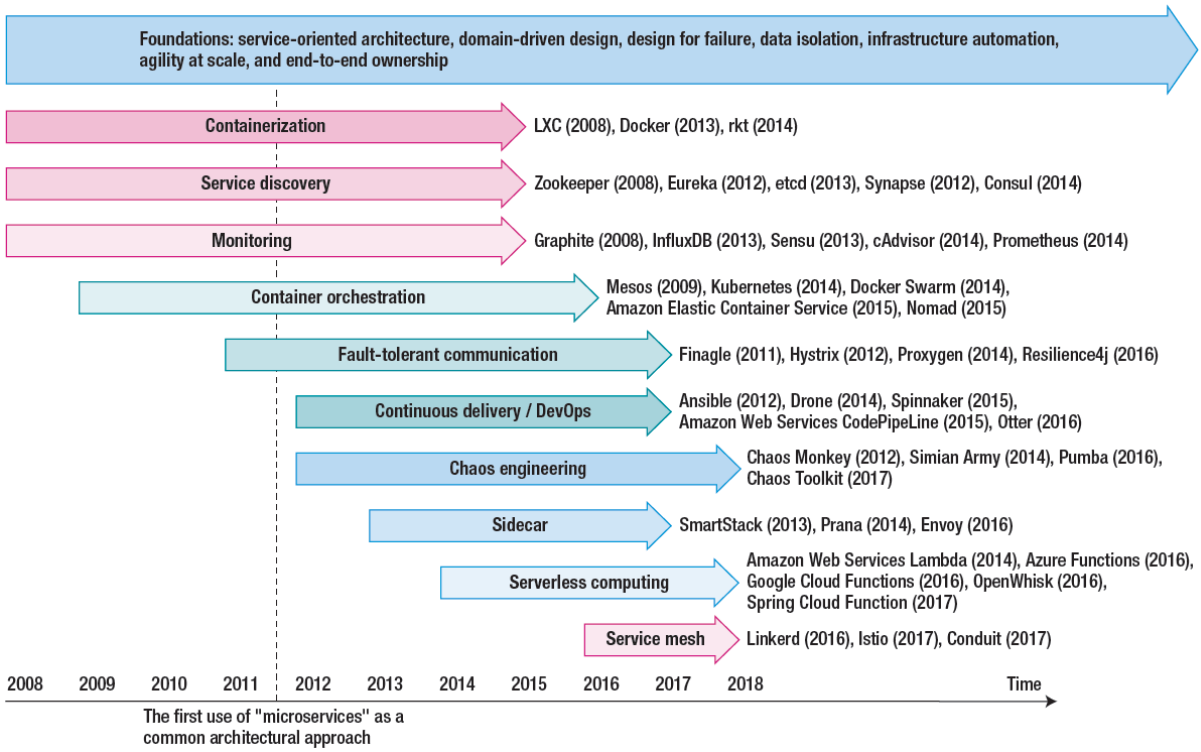
Nas últimas duas décadas, a inovação deu origem a novas arquiteturas que propuseram ótimas soluções. Nesse sentido, a AMS, que também é uma alternativa às grandes aplicações monolíticas, surgiu como um estilo arquitetural particularmente adequado às tecnologias nativas de nuvem.

De acordo com Francesco (2017), a AMS possui suas origens na AOS, porém com um grande foco em aspectos como a componentização de MS, a utilização de práticas ágeis

para o desenvolvimento, a implantação e testes de serviços, a automação com a entrega contínua e a descentralização do gerenciamento de dados e da governança de serviços.

As primeiras aplicações de MS foram fortemente influenciadas por uma nova geração de ferramentas de desenvolvimento, de implantação e de gerenciamento de *software*. À medida que a AMS se tornou mais popular, essas ferramentas continuaram evoluindo, levando a criação de sistemas ainda mais complexos. A Figura 4 mostra uma linha do tempo com dez "ondas" de tecnologias de *software*, que influenciaram a AMS no desenvolvimento, na implantação e na operação ao longo das últimas décadas.

Figura 4 - Linha do tempo das tecnologias da AMS

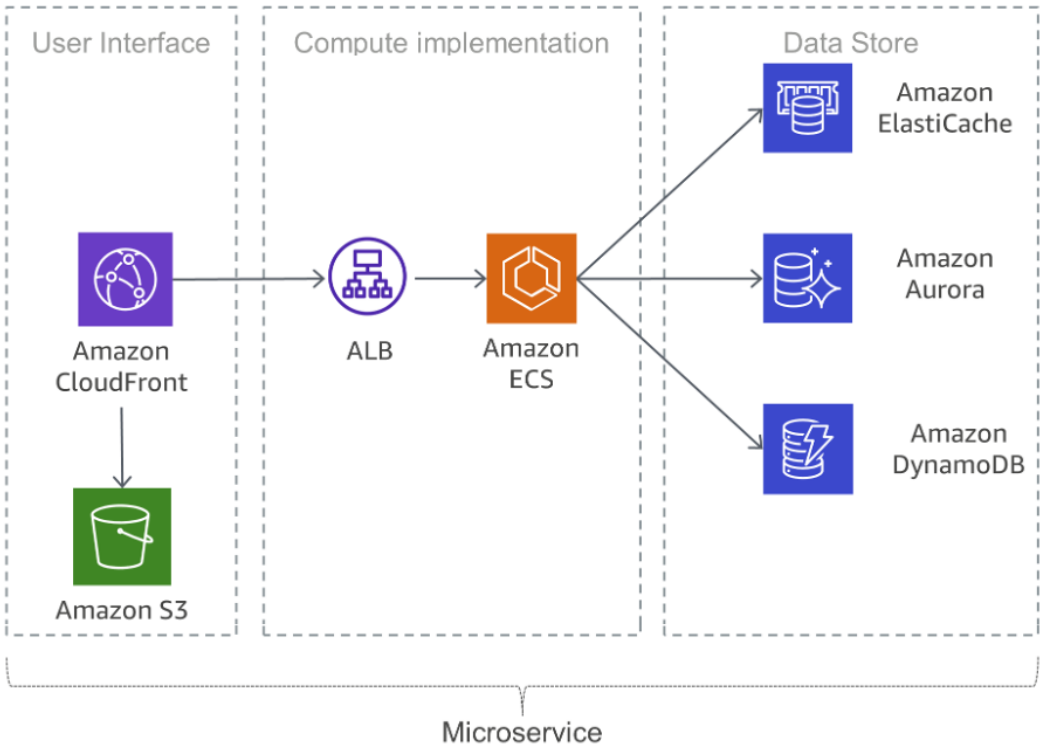


Fonte: Jamshidi et al. (2018).

Nessa abordagem, segundo Lewis e Fowler (2014), a aplicação é composta por pequenos serviços autônomos, com propósito único e bem definido, construídos conforme regras de negócio e que trabalham em conjunto. Os microsserviços comunicam-se por meio de mecanismos leves como *Application Program Interface (API)* e *Hypertext Transfer Protocol (HTTP)*, admitindo a utilização de diferentes tecnologias e linguagens de programação. Devido ao seu tamanho, são mais fáceis de compreender e manter. Devido à sua autonomia, podem ter seu próprio banco de dados, sendo implantados e escalados de forma independente e automática, além de serem tolerantes a falhas, pois uma mudança em um serviço não afetará outros.

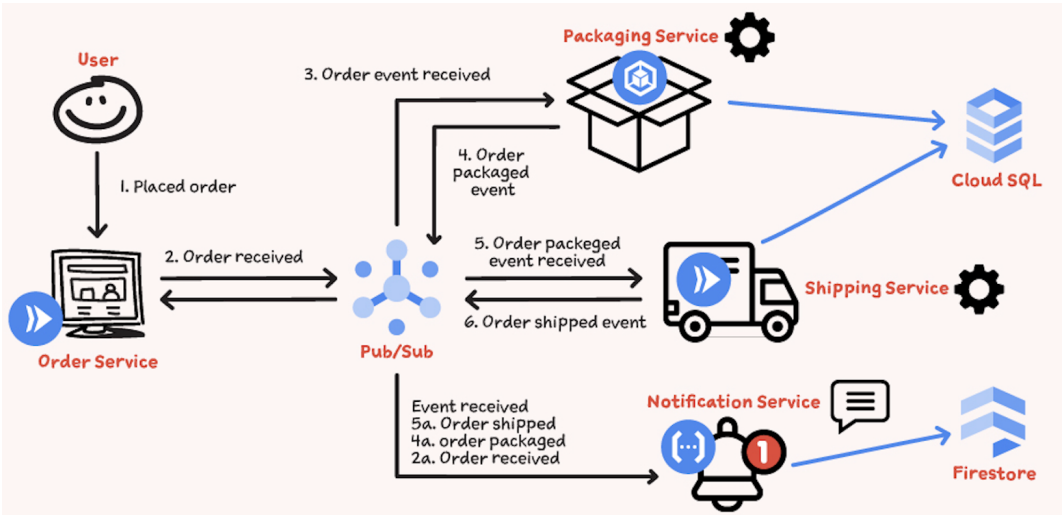
Segundo Leon et al. (2020), a AMS vem se tornando parte do processo decisório nas áreas técnica, financeira e de marketing. As Figuras 5, 6 e 7 exemplificam a AMS:

Figura 5 - Um exemplo de arquitetura de microsserviços pela AWS (*Amazon Web Services*).



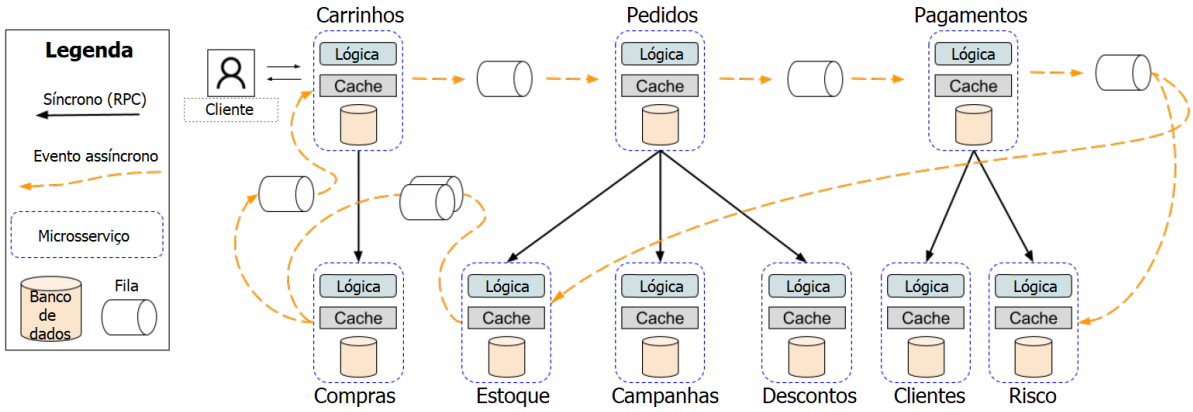
Fonte: Adaptado de AWS (2022).

Figura 6 - Exemplo da arquitetura de microsserviços pela Google Cloud.



Fonte: Adaptado de Cloud (2021).

Figura 7 - Exemplo da arquitetura de microsserviços.



Fonte: Adaptado de Laigner et al. (2021)

1.3 Comparação entre AMO, AOS e AMS

Conforme Kazanavičius e Mažeika (2019), a AMS é a melhor escolha para aplicações complexas, que estejam em evolução. Assim, uma AMO deve ser modernizada para a AMS quando:

- A AMO tornou-se muito grande e complexa, afetando a manutenibilidade;
- A modularidade e a descentralização são aspectos importantes; e
- Os benefícios de longo prazo forem preferíveis aos de curto prazo.

De acordo com Munaf et al. (2019), a AOS pode ser vista como uma AMO da perspectiva de implantação, enquanto a AMS fornece a implantação independente com a capacidade de escalabilidade dinâmica. Além disso, o gerenciamento de comunicações entre serviços na AOS é centralizado, enquanto que na AMS é distribuído.

Segundo Kazanavičius e Mažeika (2019), a AMO não está em desuso, e a AMS não é a mais adequada para qualquer contexto. A escolha dessa arquitetura somente pelo crescimento de sua adoção entre as organizações é arriscada e por esse motivo deve ser bem avaliada por arquitetos quanto à sua necessidade, já que sua escolha envolve investimento financeiro. Assim, a AMS não deve ser considerada como uma "bala de prata" para os desafios da engenharia de *software* e para as várias arquiteturas de *software* (MUNAF et al., 2019).

A Tabela 1 apresenta uma comparação entre as arquiteturas abordadas neste capítulo.

Tabela 1 - Comparação entre as arquiteturas: AMO, AOS e AMS.

Comparação	**AMO**	**AOS**	**AMS**
Acoplamento	Forte	Fraco	Fraco
Banco de dados	Compartilhado	Compartilhado	Exclusivo
Componentização	Módulo	Serviço	Microsserviço
Comunicação	Não se aplica	Centralizada (barramento)	Distribuída (mecanismos leves)
Elasticidade	Um único ponto de falha	Nenhum ponto único de falha	Nenhum ponto único de falha
Escalabilidade	Não escala sob demanda	Sob demanda, porém comparativamente menos escalável em relação à AMS	Sob demanda
Equipes	Não se aplica	Múltiplas (grandes)	Individuais (pequenas)
Implantação	Todo o sistema	Independente	Individual (por microsserviço)
Gerenciamento	Não se aplica	Centralizado	Distribuído
Granularidade	Unidade única	Grossa	Fina
Tecnologia	Homogênea	Homogênea	Heterogênea

Fonte: Adaptado de Munaf et al. (2019), Liu et al. (2020).

Com o objetivo de analisar e organizar o complexo domínio da AMS, pode-se utilizar técnicas de análise de domínio para a identificação de suas características.

1.4 **Modelo de características**

1.4.1 Visão geral

Segundo Lee, Kang e Lee (2002), a modelagem de características (MC) é uma das técnicas mais populares de análise de domínio que consiste na descoberta e descrição de características comuns e distintivas, bem como suas relações estruturais, que são visíveis ao usuário final a quem o modelo se destina. As relações entre as características em uma família de produtos geralmente são capturadas em gráficos chamados de modelos de características (do inglês *Feature Models*) (ALTURKI; KHÉDRI, 2010).

Um dos métodos de análise de domínio, conhecido como *Feature Oriented Domain Analysis* (FODA), criado por Kang et al. (1990), prevê a modelagem de *features* para a identificação de partes comuns e variáveis de um produto, com o objetivo de capturar o entendimento dos usuários finais e clientes a respeito das capacidades gerais de aplicações

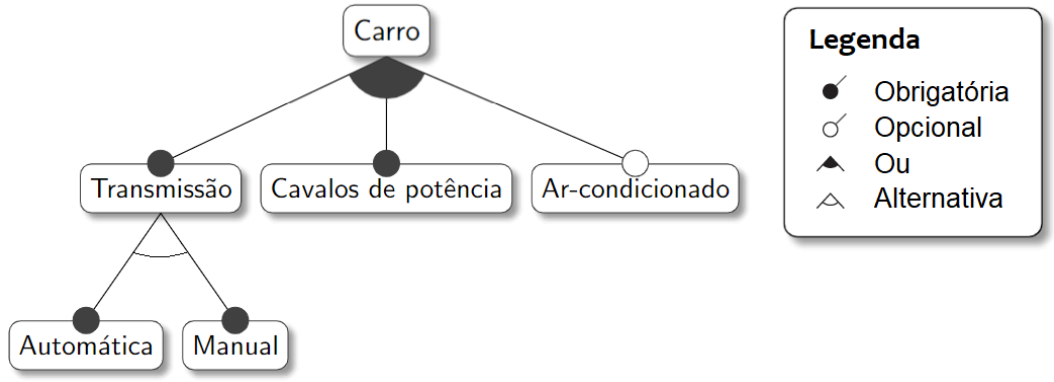
em um domínio. O modelo prevê o agrupamento lógico de funcionalidades de mesmo interesse, podendo gerar relacionamentos obrigatórios, alternativos ou opcionais.

De acordo com Kang et al. (1990), uma funcionalidade (*feature*) pode ser classificada em:

- **Obrigatória:** a funcionalidade deve estar presente em todos os produtos;
- **Opcional:** a funcionalidade pode ou não estar presente em um produto;
- **Ou:** A relação entre uma característica pai e suas características filhas é categorizada como "Ou" se pelo menos uma das características filhas deve ser selecionada. As características "Ou" são representadas por um arco preenchido que agrupa as características filhas; e
- **Alternativa:** A relação entre uma característica pai e suas características filhas é categorizada como "alternativa" se uma (e apenas uma) das características filhas deve ser selecionada. As características "alternativas" são representadas por um arco vazio que agrupa as características filhas.

A Figura 8 apresenta um modelo de características do domínio automobilístico, no qual as características "Automática" e "Manual", que se referem ao tipo de "Transmissão", são alternativas e, por isso, mutuamente exclusivas, enquanto "Transmissão" e "Cavalos de potência" são obrigatórias. Já a característica "Ar-Condicionado" é opcional.

Figura 8 - Exemplo mostrando características de um carro.



Fonte: Adaptado de Kang et al. (1990).

1.4.2 Características Não-Funcionais (CNF) em Arquitetura de Software

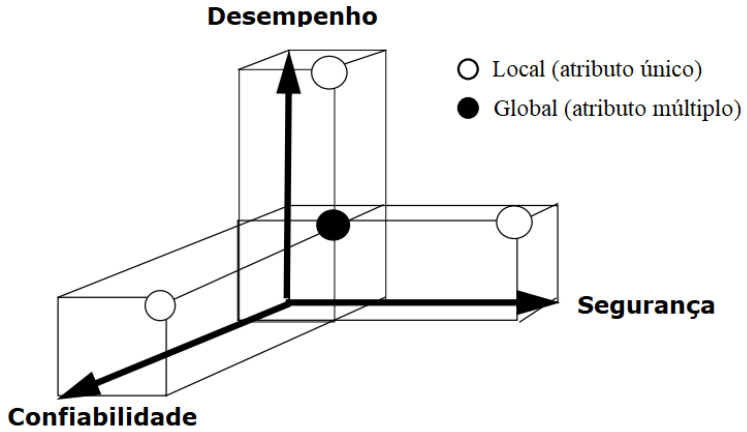
A estrutura geral do sistema (a solução), definida pela arquitetura de *software*, está intrinsecamente voltada para atender aos requisitos funcionais e satisfazer aos requisitos essenciais de qualidade (o problema) (BOUCKÉ; HOLVOET, 2005). Interesses (*concerns*) em sistemas de *software* são questões importantes para os (*stakeholders*).

Requisitos não-funcionais são as necessidades de um sistema para realizar suas operações sob um conjunto de restrições e expectativas de qualidade. Além disso, identificar requisitos não-funcionais e relacioná-los com interesses (*concerns*) arquiteturais é mais difícil do que lidar com requisitos funcionais, uma vez que os usuários possuem mais dificuldade em explicar os requisitos não-funcionais durante a modelagem de requisitos (GOKYER et al., 2008).

Há uma distinção entre requisitos e interesses arquiteturais: os requisitos descrevem o problema, enquanto os interesses arquiteturais estão relacionados à solução. Os interesses arquiteturais não são apenas influenciados pelos requisitos, mas também pela estratégia de solução. Com isso, um único interesse arquitetural contribui, positiva ou negativamente, para o atendimento de vários requisitos (BASS; CLEMENTS; KAZMAN, 2003).

Consequentemente, os arquitetos de *software* devem analisar as compensações entre os vários atributos conflitantes para satisfazer os requisitos (BARBACCI et al., 1995). A Figura 9 apresenta uma negociação (*trade-offs*) entre atributos de qualidade de *software*.

Figura 9 - *Trade-offs* entre atributos de qualidade de *software*.



Fonte: Adaptado de Barbacci et al. (1995).

A arquitetura possui uma importância crítica para o sucesso de projetos de *software* e ela só é eficaz quando é documentada, permitindo aos *stakeholders* entender e usar a arquitetura conforme o planejamento. No entanto, segundo Gorton (2011) a documentação da arquitetura é uma questão problemática, sendo muito comum a ausência ou a insuficiência dela em alguns projetos de *software*. Às vezes, quando há alguma documentação,

ela é obsoleta, inadequada e pouco útil.

No outro extremo, o excesso de informação na documentação da arquitetura pode ocasionar os mesmos problemas mencionados. Dessa forma, a documentação efetiva de uma arquitetura é tão importante quanto criá-la, pois se a arquitetura não é compreendida, não atenderá aos objetivos como uma visão unificadora para o desenvolvimento de *software* (CLEMENTS et al., 2003).

No escopo desse trabalho, uma característica deve ser entendida como um aspecto que, quando não considerado, pode impactar na arquitetura do *software* de forma que esta precise ser reconstruida.

1.5 **Visualização de dados**

A necessidade de visualização foi reconhecida pela primeira vez nas ciências durante o final da década de 1980, quando o poder crescente da computação e o custo decrescente do armazenamento digital criaram um aumento na quantidade e na complexidade dos dados que precisavam ser gerenciados, processados e compreendidos (FULLER, 2008).

A visualização de dados é um termo relativamente recente, que expressa uma ideia mais abrangente do que apenas uma representação de dados na forma de gráfico. As informações por trás dos dados devem ser reveladas por meio de uma exibição apropriada (CHEN; HÄRDLE; UNWIN, 2007).

Segundo Unwin (2020), a visualização de dados representa uma das ferramentas fundamentais da ciência de dados moderna. O objetivo principal é promover a visualização de dados e de estatísticas, com a interpretação das exibições para obtenção de informações. A eficácia da visualização depende da capacidade de representação de informações de forma clara e precisa e da capacidade de interação humana para descobrir o significado da informação (FEW, 2009 apud OLIVEIRA, 2015).

A visualização de dados utilizando *layouts* no formato em árvore pode ser uma estratégia para representar modelos de características, sendo possível representar as características e suas relações de generalização e de especialização. *Layouts* no formato em árvore são úteis para expressar relacionamentos hierárquicos (MEEKS, 2015).

1.6 **Considerações finais**

Este capítulo apresentou conceitos essenciais sobre arquitetura de *software*, além da comparação entre AMO, AOS e AMS. Também foram elucidados conceitos pertinentes à MC, às CNF em arquitetura de *software* e à visualização de dados. No próximo capítulo, são apresentados trabalhos relacionados ao problema tratado nessa dissertação.

2 TRABALHOS RELACIONADOS

Neste capítulo, com o objetivo de entender e identificar as características relevantes da AMS, são analisadas diversas iniciativas relacionadas a esse tema. Assim, os trabalhos relacionados são apresentados em duas categorias: (i) estudos sobre métodos e técnicas de avaliação arquitetural e (ii) estudos sobre classificações e taxonomias de AMS.

2.1 Estudos sobre métodos e técnicas de avaliação arquitetural

A seguir são apresentados trabalhos sobre métodos e técnicas de avaliação arquitetural.

Mattsson, Grahn e Mårtensson (2006) apresentam um *survey* sobre métodos de avaliação de arquitetura de *software*. O estudo teve como foco métodos para avaliar um ou vários atributos de qualidade como desempenho, manutenibilidade, testabilidade e portabilidade. Eles descobriram que a maioria dos métodos de avaliação aborda apenas um atributo de qualidade e poucos podem avaliar vários atributos simultaneamente utilizando o mesmo *framework* ou método. Além disso, foi identificado que muitos métodos são utilizados e validados apenas pelos próprios autores.

Barcelos (2006) identifica e caracteriza as principais abordagens de avaliação arquitetural descritas na literatura e realiza uma revisão sistemática. As abordagens foram avaliadas de acordo com os critérios: extensibilidade, simplicidade, generalidade e adaptabilidade. No entanto, nenhuma abordagem atendeu a essas características simultaneamente. Segundo o autor, o fato se deve à quatro problemas principais presentes nessas abordagens: grande subjetividade das abordagens, elevado custo de aplicação, dificuldades para avaliar simultaneamente o atendimento a várias características de qualidade e contexto limitado para a aplicação de algumas abordagens.

Lehmann e Sandnes (2017) propõem um *framework* de avaliação para auxiliar arquitetos de *software* na seleção de uma estratégia e de um conjunto de tecnologias para implementar a entrega contínua em um sistema de *software* baseado em MS. A estratégia foi definida com a combinação de componentes tecnológicos e medidas organizacionais adotadas para alcançar a entrega contínua. O *framework* é baseado em experiências documentadas na literatura existente, bem como na indústria.

Engel et al. (2018) apresentam um método de avaliação de AMS baseada em princípios arquiteturais identificados na literatura e na indústria, como o tamanho reduzido dos serviços, a escalabilidade, o acoplamento fraco e a performance. Os Princípios e métricas para a avaliação do *design* arquitetural foram derivados com base em um estudo que apresenta os desafios atuais nas AMS. Os dados arquiteturais necessários são captu-

rados por meio de uma abordagem de engenharia reversa a partir de rastros de dados de comunicação.

Cardarelli et al. (2019) propõem uma abordagem para avaliação de atributos de qualidade de *software* para a arquitetura de sistemas baseados em MS. A abordagem permite a desenvolvedores produzir modelos de arquitetura do sistema por meio de técnicas de recuperação, contribuir para um ecossistema especificado e automaticamente computável de atributos de qualidade de *software* para AMS, e medir e avaliar continuamente a arquitetura de seus sistemas através da utilização dos atributos de qualidade de *software* definidos no ecossistema. A abordagem é implementada utilizando técnicas de engenharia orientada a modelos.

Cojocaru, Oprescu e Uta (2019) apresentam um conjunto de critérios de avaliação de qualidade de MS resultantes da aplicação de técnicas de decomposição semiautomática de aplicações monolíticas utilizados na literatura, considerando tanto a AMS quanto à AOS, conforme mostra a Figura 10.

Rosa et al. (2020) apresentam um método sistemático para análise de *trade-offs* arquiteturais com base em padrões. Eles conduziram um estudo para identificar padrões arquiteturais de MS que influenciam decisões de *design* estrutural relacionadas ao tamanho dos serviços, ao compartilhamento de banco de dados e ao nível de acoplamento de serviços. Consideraram, no geral, que o método ajuda arquitetos de *software* na identificação de padrões arquiteturais mais apropriados para um determinado contexto.

Figura 10 - Visão geral dos critérios de qualidade.

	Quality Attributes																										
	Static Analysis					Dynamic Analysis																					
Citations	Granularity	LOC	Open Interface	High Cohesion	Loose Coupling	Execution Cost	Response Time	Availability	Succ. Exec. Rate	Usage Frequency	Scalability	Independence	Maintainability	Deployment	Health Management	Modularity	Manageability	Performance	Reusability	Tech. Heterogeneity	Agility	Security	Load Balancing	Org. Alignment
[9]						⊕	⊖	⊖	⊕	⊖	⊖														
[10]			⊖		⊖						⊖	⊕	⊕	⊕	⊕	⊖	⊕	⊖	⊕	⊖	⊖	⊕	⊖	⊖
[11]	⊕		⊕	⊕	⊕						⊕	⊕	⊖	⊖	⊖	⊖	⊕	⊕		⊖		⊖		⊖
[12]		⊕		⊕																				⊖
[13]	⊕	⊕			⊕	⊖	⊕							⊕	⊕			⊕						⊖
[14]	⊕											⊖		⊖						⊖				⊖
[15]													⊖	⊕		⊖	⊕					⊕	⊖	
[16]	⊕			⊖	⊖						⊖	⊕		⊖						⊖				
[17]				⊕	⊕						⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖			⊖	⊖	⊖
[18]	⊖			⊖	⊖							⊖	⊖	⊖				⊖			⊖			⊖
[19]			⊖		⊖		⊕	⊖			⊕			⊕	⊖			⊕				⊕	⊖	

Criteria	Meaning
Granularity	Size of microservice
LOC	Lines of code
Open Interface	Accessible and standardized interface
Succ. Exec. Rate	Rate of successful executions
Tech. Heterogeneity	Heterogeneity regarding the programming language
Org. Alignment	Alignment with organization's communication patterns

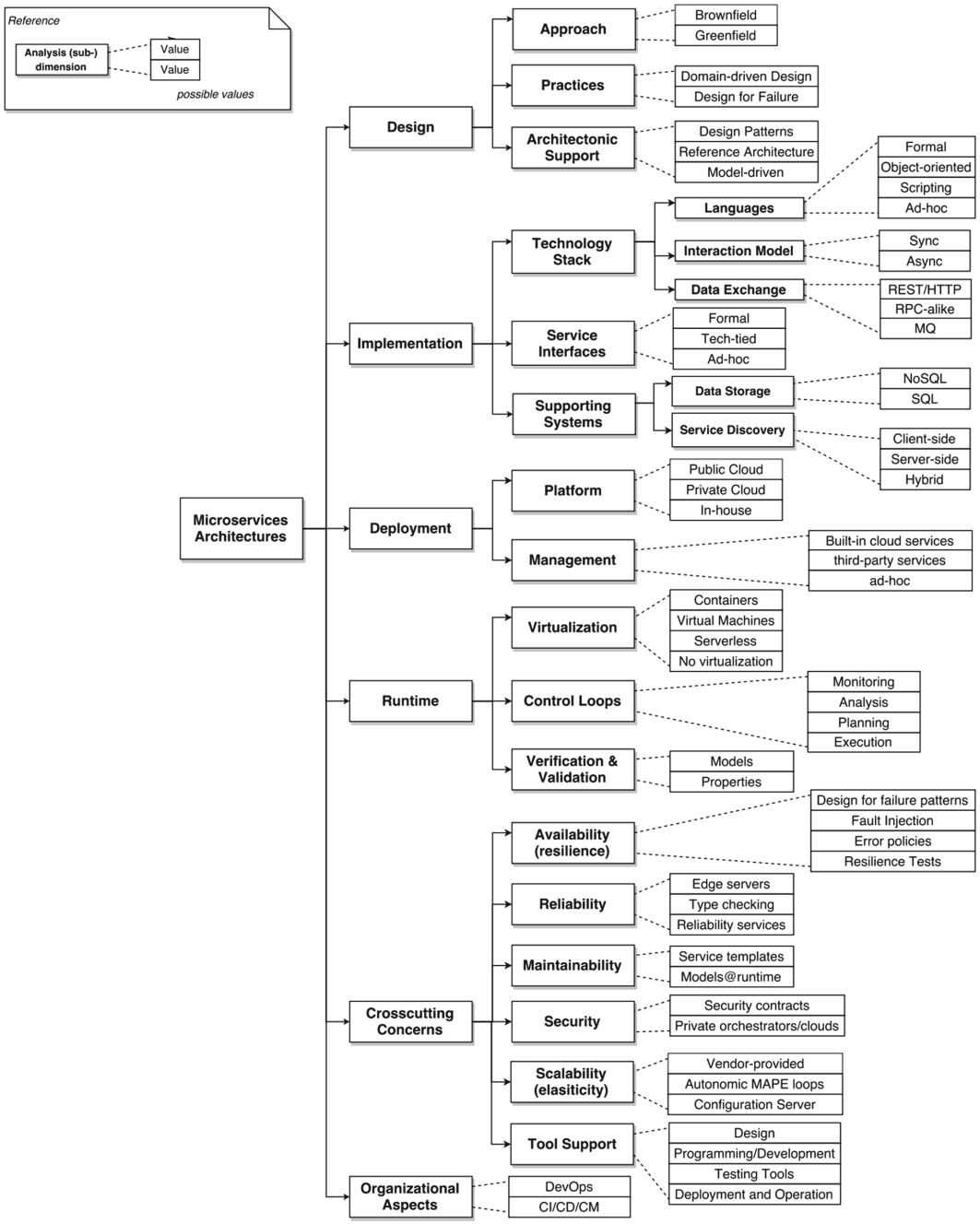
The presence of workload description is indicated by ⊕, its absence by ⊖.

Fonte: Adaptado de Cojocaru, Oprescu e Uta (2019).

2.2 **Estudos sobre classificações e taxonomias de AMS**

Garriga (2018) apresenta uma análise preliminar de um *framework* que captura a compreensão fundamental das AMS na forma de taxonomia, abrangendo o ciclo de vida e também aspectos organizacionais, conforme mostra a Figura 11.

Figura 11 - Taxonomia de microsserviços.

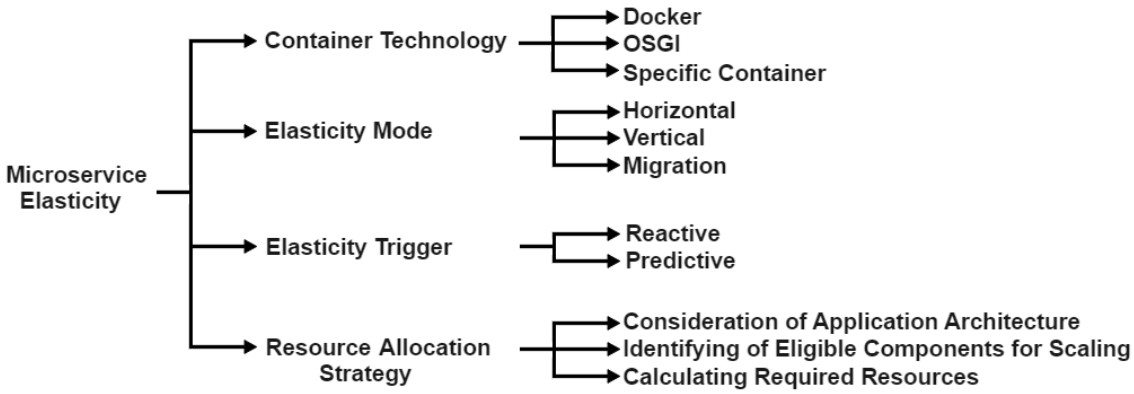


Fonte: Adaptado de Garriga (2018).

Garriga (2018) enriqueceu sua proposta tomando como base a experiência adquirida em *frameworks* de classificação no contexto de AOS, tanto para *Web Services* (GARRIGA et al., 2015), quanto para serviços *RESTful* (GARRIGA et al., 2016).

Fourati, Marzouk e Jmaiel (2021) propõem uma classificação para a elasticidade na AMS, com base em quatro critérios: *Container Technology*, *Elasticity Mode*, *Elasticity Trigger* e *Resource Allocation Strategy*. Trata-se de uma visão geral sobre soluções de escalonamento automático durante o tratamento da elasticidade, conforme a Figura 12.

Figura 12 - Elasticidade em microsserviço.



Fonte: Adaptado de Fourati, Marzouk e Jmaiel (2021).

Söylemez, Tekinerdogan e Tarhan (2022a) realizaram uma revisão sistemática da literatura e identificaram nove categorias básicas de desafios encontrados na adoção da AMS. O estudo apresenta orientações sobre soluções para cada desafio identificado. A Tabela 2 apresenta as nove categorias de problemas que foram identificadas.

Tabela 2 - Problemas identificados na adoção de AMS em estudos primários.

#	**Problemas**
P1	Service Discovery
P2	Data Management and Consistency
P3	Testing
P4	Performance Prediction, Measurement and Optimization
P5	Communication and Integration
P6	Service Orchestration
P7	Security
P8	Monitoring, Tracing and Logging
P9	Decomposition

Fonte: Adaptado de Söylemez, Tekinerdogan e Tarhan (2022a).

A Tabela 3 apresenta a categoria de problemas com as respectivas orientações de solução.

Tabela 3 - Orientações para os desafios identificados na AMS.

Desafio	**Orientação da solução**
P1.Service Discovery	- Client-Side Service Discovery. Study J - Server-Side Service Discovery. Study J - Service Registry. Study J - ICN-Based Service Discovery. Study I - Static–Dynamic Service Description. Study AF - Stateful Routing Mechanism. Study M
P2. Data Management and Consistency	- Multi Agent-Based Framework. Study S - BAC Theorem for Backing up Data. Study AH - Solution Framework. Study C
P3.Testing	- Reusable BDD-Based Acceptance Test Architecture. Study B - A Flow for Regression Test. Study AG - An Architecture and Framework to Automate Performance Test. Study G - A Framework for Testing the Failure-Handling Capabilities. Study H - Validation Framework. Study A - Automation Testing Framework. Study AL
P4.Performance Prediction, Measurement and Optimization	- Simulation Model. Study AA - Performance Model and Prediction Method. Study AK - Performance Prediction Model. Study AR - Performance Analytical Model. Study E - An approach for the Quantitative Assessment of Microservice Architecture Deployment Configuration Alternatives. Study BB - Performance Degradation Prediction Framework. Study BY - A Model-Driven Approach for Continuous Performance Improvement. Study CF
P5.Communication and Integration	- Reference Architecture and Orchestrator Language. Study AI - High-performance Userspace Networking Solution. Study R

continuação na próxima página

P6.Communication and Integration	- An Extendable Solution for Autoscaling. Study L - Database-is-the-Service Pattern. Study F - Workflow Scheduling Algorithm. Study AV - Autoscaling Research Pipeline. Study BF - Ant Colony Algorithm for Microservice Scheduling Optimization. Study AN - A Novel Scheduling Strategy. Study BA - A Lightweight and Flexible System for Autoscaling. Study AX - A Generic Architecture and Implementation for Automated Orchestration. Study AU - Configuration Models and Tool for Analyzing the Availability. Study W - Storage Service Orchestrator Framework. Study Y - A Monitoring-Based Architecture for Managing Deployment. Study AM - Decentralized Orchestrator. Study AT - Process Definition for An Elasticity Controller. Study AE - Decentralized Load Balancing Algorithm. Study N - Overload Control Method. Study Q - A Hybrid Approach Combining Client-side and Server-side Load Balancing. Study BE - Queue-Based Chain-Oriented Load Balancing Method. Study AW - A Novel Fair Weighted Affinity-based Scheduling Approach. Study O - A Novel Scheduling Framework for Kubernetes. Study AQ - Dynamic Microservice Scheduling Algorithm for Mobile Edge Computing. Study AY - Resource Allocation Optimization Approach. Study AO - Many-Objective Genetic Algorithm Scheduler. Study BD - A Novel Formula andModel for Determining the Thresholds of Total Resource Consumption. Study BH - Autoscaling Research Pipeline. Study BI - Using Declarative Business Processes for Service Orchestration. Study BK - RL agent based intelligent autoscaling model. Study BM - A Decision Framework to Select Right Microservice Collaboration Pattern. Study BP - Elastic Scheduling Algorithm. Study BW - Layered Container Structure for Microservice Deployment. Study BX - Autoscaling Framework for Microservice chain. Study BZ - Dynamic Flow Control Algorithm. Study CA - Microservice Rescheduling Framework. Study CB - A Kubernetes Controller for Managing Availability. Study CD
P7.Security	- An Approach That Provides Authentication and Decentralized Role-Based Authorization. Study T - A Platform for Identity and Access Control of Microservices. Study AC - Access Control Optimization Model. Study AJ - Prototype Layered Security Framework (Hardware, Virtualization, Cloud, Communication, Application, and Orchestration). Study V - An Approach for Handling Security as Security-as-a-Service. Study D - Extended Role-Based Access Control Model. Study BT

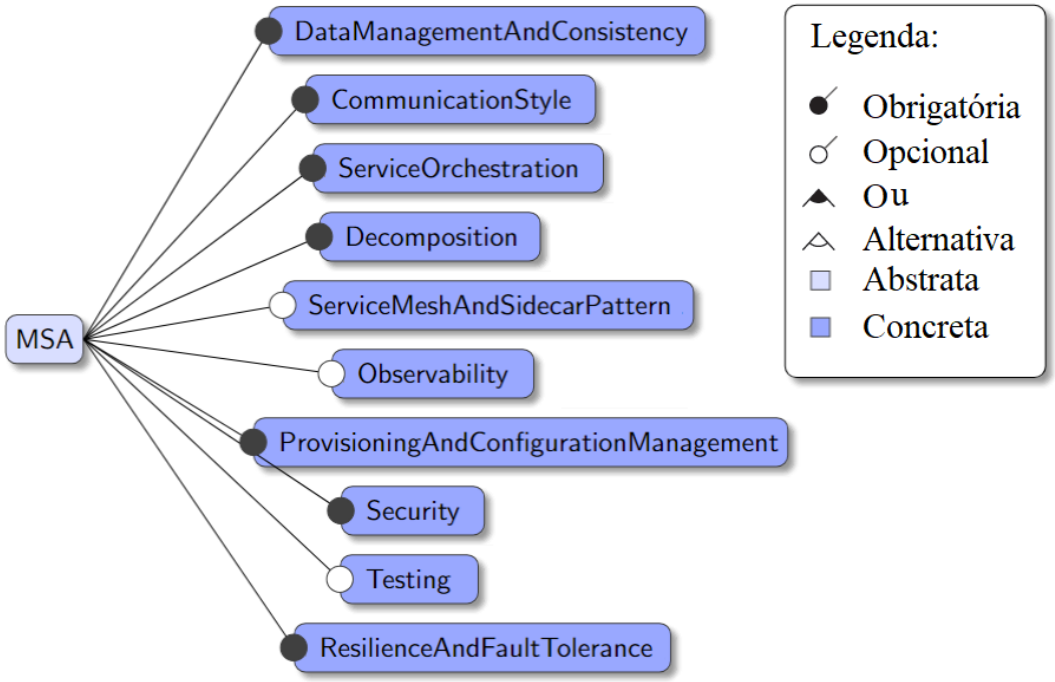
continuação na próxima página

P8.Monitoring, Tracing and Logging	- An Approach and Tool for Generating Service Dependency Graph. Study X - A Tool for Generating Service Causal Graph. Study Z - An Approach For Analyzing Architecture. Study AD - A Tool For Handling Traversing Distinct Type of Logs. Study AS - A Tool For Architecture Recovery. Study P - A Root Cause Analysis Framework for Detecting Anomalies. Study AZ - An Execution Trace-Based Root Cause Location Method. Study BG - A Graph-based Trace Analysis Approach—Study BJ - An Offline Approach to Distributed Tracing. Study BL - A Four Layered Framework for Detection and Diagnosis of FaultyMicroservices. Study BN - In-Kernel Transparent Monitoring Service. Study BO - Microservice Fault Detection Method Based on Correlation Analysis. Study BQ - A Fault Model-Based Root Cause Localization Framework. Study BR - An Anomaly Detection Method based on Semi-Supervised Learning. Study BS - A Root Cause Localization Approach. Study BU - Lightweight Spectrum-Based Performance Diagnosis Tool. Study BV - An Anomaly Detection Approach with Execution Trace Comparison. Study CE - An Agent-Based Monitoring Platform to Detect Anomalies and Unexpected System Dependencies. Study CG
P9.Decomposition	- A Conceptual Methodology for Deciding Right Size of Microservices. Study AB - A Functional Decomposition Approach. Study U - A Dataflow-Driven Decomposition. Study AP - A Dependency Capturing and Clustering-Based Microservice Identification Approach. Study CC

Fonte: Adaptado de Söylemez, Tekinerdogan e Tarhan (2022a).

Söylemez, Tekinerdogan e Tarhan (2022b) apresentam um *framework* de caracterização, no qual as características chaves e desafios foram identificados e sintetizados com base nas principais soluções da AMS providas pelos principais fornecedores, conforme a Figura 13.

Figura 13 - Características de primeiro nível do diagrama de características da AMS.



Fonte: Adaptado de Söylemez, Tekinerdogan e Tarhan (2022b).

Os autores realizaram um mapeamento entre as características e as respectivas tecnologias da AMS, conforme a Tabela 4.

Tabela 4 - Mapeamento de características com as tecnologias da AMS.

Característica	**Tecnologia/Produto/Serviço**
Testing/Chaos Engineering	- Chaos Monkey - Chaos Toolkit - Simian Army
Testing/Service Component Test	- Spring Cloud Contract Test
Resilience and Fault Tolerance/Circuit Breaker	- Netflix Hystrix - Resilience4j
Communication Style/API Gateway	- Nginx - Netflix/Zuul - Spring Cloud Gateway
Communication Style/Domain Spesific Protokol	- SMTP - IMAP
Communication Style/Async Communication	- Apache Kafka - Rabbit MQ - Active MQ
Observability/Log Analysis	- Kibana - Datadog - LogDNA

continuação na próxima página

Observability/Distributed Tracing	- Zipkin - Datadog - OpenCensus - Sentry - LogDNA
Observability/Monitoring	- Prometheus - Graphite - Grafana - InfluxDB - Zabbix
Observability/Log Aggregation	- Kibana - Datadog - LogDNA
Observaility/Exception Tracking	- Sentry
Provisioning and Configuration Management	- Ansible - Chef - Puppet - SaltStack
Provisioning and Configuration Management/Infrastructure as Code	- Terraform
Security/Authentication	- CAS - Spring Security - SSO
Security/Authorization	- JWT - Spring Security
Decomposition/Decompose by Subdomain	- Domain Driven Design
Service Orchestration	- Kubernetes - Apache Mesos + Marathon - Docker Swarm
Service Mesh and Sidecar Pattern	- Istio - Linkerd - Envoy - Redhat Openshift
Deployment/CI & CD	- Jenkins - CircleCI - Travis - DroneCI - Gitlab CI - Bamboo
Deployment/Container	- Docker - LXC
Deployment/Virtual Machine	- VMWare - VirtualBox
Load Balancing/Server-side	- Nginx - Zuul - Eureka

continuação na próxima página

Load Balancing/Client-side	- Ribbon Client
Service Discovery/Service Registry	- Eureka - Zuul - Consul - Apache Zookeeper
Service Discovery/Server-side	- Eureka - Zuul - Consul - Apache Zookeeper
Service Discovery/Client-side	- Ribbon Client

Fonte: Adaptado de Söylemez, Tekinerdogan e Tarhan (2022b).

Além disso, os mesmos autores também realizaram uma comparação entre as características e os serviços fornecidos pelas empresas Amazon Web Services, Google Cloud e Microsoft Azure, conforme a Figura 14.

Figura 14 - Comparação de características baseadas em serviços entre AWS, Google Cloud e Microsoft Azure.

Característica		**AWS**	**Google Cloud**	**Microsoft Azure**
Communication Style	Async Communication	AWS MQ AWS SNS AWS SQS AWS Kinesis	Google Dataflow Google Pub/Sub	Azure Queue Storage Azure Service Bus Azure Event Grid Azure Event Hubs
	API Gateway	AWS API Gateway	Google Apigee	Azure API Management
Service Orchestration		AWS ECS AWS EB AWS EKS	Google Cloud Run Google App Engine Google Kubernete Engine	Azure Container Instances Azure App Service Azure EKS
Service Orchestration	Deployment/CI and CD	AWS CodeDeploy AWS CodeBuild AWS CodePipeline	Google Cloud Build	Azure Devops
	Deployment/Serverless Function	AWS Lambda AWS Step Function	Google Cloud Function Google Cloud Composes	Azure Durable Azure Functions
	Deployment/Container	AWS Fargate AWS EKS	Google Cloud Run Google Kubernete Engine	Azure Container Instance Azure Kubernete Service
	Deployment/VM	AWS EC2	Google Compute Engine	Azure VM
	Auto-scaling	AWS EC2 Auto-scaling	Google Computer Engine Auto-scaling	Azure Virtual Machine Scale Set
	Load Balancing/Server-side	AWS ELB	Google Cloud Engine Load Balancing	Azure Load Balancing
	Service Discovery/Server-side	AWS Route 53 AWS Cloud Map AWS ELB	Google Cloud DNS Google Cloud Engine Load Balancing	Azure DNS Azure Load Balancing
Service Mesh and Sidecar Pattern		AWS AppMesh	Google Anthos Service Mesh	Azure Service Fabric Mesh
Observability	Log Analysis	AWS Elasticsearch Service AWS Redshift AWS Quicksight AWS Athena	Google Elasticsearch Service Google BigQuery	Azure Elasticsearch Service Azure PowerBI Azure Data Lake Analytics
	Exception Tracking	AWS CloudWatch	Google Cloud Debugger Google Cloud Trace	Azure Application Insights Azure Monitor
	Log Aggregation	AWS CloudWatch	Google Cloud Logging	Azure Application Insights Azure Monitor
	Audit Logging	AWS CloudTrail AWS Config	Google Audit Logs Google Cloud Asset Inventory	Azure Monitor
	Distributed Tracing	AWS X-Ray	Google Cloud Debugger Google CloudTrace	Azure Monitor
	Monitoring	AWS CloudWatch	Google Cloud Monitoring	Azure Monitor
Provisioning and Configuration Management		AWS CloudFormation	Google Cloud Deployment Manager	Azure Resource Manager Azure VM extensions Azure Automation
Security		AWS Cognito	Google Firebase Authentication	Azure Active Directory B2C

Fonte: Adaptado de Söylemez, Tekinerdogan e Tarhan (2022b).

2.3 Aspectos identificados a partir da análise

- Levando em consideração o *survey* realizado por Mattsson, Grahn e Mårtensson (2006) e a revisão sistemática realizada por Barcelos (2006), sobre métodos de avaliação arquitetural, verifica-se a dificuldade de realizar uma avaliação considerando um conjunto abrangente de características de forma simultânea sem o custo elevado, como ressaltado por Barcelos (2006);

- O *framework* de avaliação proposto por Lehmann e Sandnes (2017) para auxiliar arquitetos de *software* na seleção de uma estratégia e de um conjunto de tecnologias para implementar a entrega contínua em um sistema de *software* não avalia outras caracteristicas encontradas na AMS;

- O método de avaliação de AMS proposto por Engel et al. (2018) pode fornecer informações referentes aos princípios da AMS para subsidiar uma avaliação. No entanto, a abordagem pode não ser considerada abrangente diante do conjunto abrangente de características encontradas na AMS;

- A taxonomia proposta por Garriga (2018) pode contribuir na compreensão da AMS. Contudo, em relação aos objetivos do presente estudo, abrange um nível demasiado de detalhes, como o ciclo de vida dos MS, aspectos organizacionais, linguagens e ferramentas;

- A abordagem proposta por Cardarelli et al. (2019) possui foco na avaliação de um conjunto restrito de atributos de qualidade. Além disso, a abordagem apresenta a dependência de fontes de informação específicas (*e.g.*, arquivos Docker);

- O conjunto de atributos de qualidade aplicáveis a MS proposto por Cojocaru, Oprescu e Uta (2019) pode fornecer informações para subsidiar uma ferramenta de avaliação de AMS.

- O método sistemático para análise de *trade-offs* arquiteturais com base em padrões proposto por Rosa et al. (2020) não foi avaliado por pesquisadores e arquitetos de *software*;

- A classificação proposta por Fourati, Marzouk e Jmaiel (2021), que utiliza somente uma característica, tem pouca contribuição diante do conjunto abrangente de características encontradas na AMS. Além disso, em sua classificação, o autor faz referência a tecnologias específicas, o que, com o tempo, poderá tornar-se defasada; e

- Em relação às propostas dos trabalhos de Söylemez, Tekinerdogan e Tarhan (2022a) e de Söylemez, Tekinerdogan e Tarhan (2022b), ambas podem ser utilizadas como

fonte de consulta; a primeira, fornecendo soluções para as nove categorias de desafios encontrados na adoção da AMS, e a segunda, com o mapeamento das características com as tecnologias e os serviços da AMS.

Conforme mencionado em capítulos anteriores, um desafio enfrentado por arquitetos de *software* é ter uma compreensão clara das características desejadas em um sistema e dos recursos necessários para implementá-las.

Desse modo, a avaliação de um microsserviço sob a perspectiva de diversas características pode revelar a necessidade de priorização de algumas em detrimento de outras, podendo implicar, inclusive, na necessidade de contratação de produtos ou serviços externos para suportar tais necessidades. Para isso, é fundamental que essas características sejam bem compreendidas e que essas informações estejam registradas em um documento junto à arquitetura de *software*.

Além disso, existem problemas envolvendo o dimensionamento da infraestrutura necessária para atender às características: o subdimensionamento, ocasionando a necessidade de expansão ou substituição da mesma; o superdimensionamento, com a ociosidade de recursos não utilizados; e a própria decisão de comprar ou alugar a infraestrutura.

Esses problemas podem ocasionar falhas de segurança, vazamento de informações, perda de dados, lentidão ou indisponibilidade na utilização dos serviços, causando não só danos relacionados à imagem de uma organização, como também perdas financeiras.

Com base na análise realizada, um modelo de características da AMS pode servir como um ponto de partida para a organização desse domínio. Isso, por sua vez, pode subsidiar o desenvolvimento de uma ferramenta destinada à análise e avaliação de MS, com o propósito de auxiliar arquitetos e desenvolvedores de *software* na compreensão desses interesses não-funcionais, contribuindo com os processos de desenvolvimento e manutenção de MS.

O capítulo a seguir apresenta abordagem utilizada para atingir os objetivos deste trabalho.

3 ABORDAGEM PROPOSTA

A partir da revisão da literatura descrita no Capítulo 2 deste trabalho, observa-se a importância da avaliação de arquiteturas de *software* e da compreensão das diversas características envolvidas na AMS. Nesse contexto, esse capítulo apresenta a metodologia e as atividades necessárias para se atingir os objetivos deste trabalho, além da estratégia e dos critérios utilizados para subsidiar a proposta inicial do modelo de características da AMS. Por fim, é apresentada uma proposta de utilização de técnica de visualização com o objetivo de expressar o grau de atendimento das características da AMS.

3.1 Visão geral

O presente trabalho tem como objetivo:

(i) prover um **modelo de características** sobre a AMS;

(ii) prover um **glossário de termos** com a definição de cada característica;

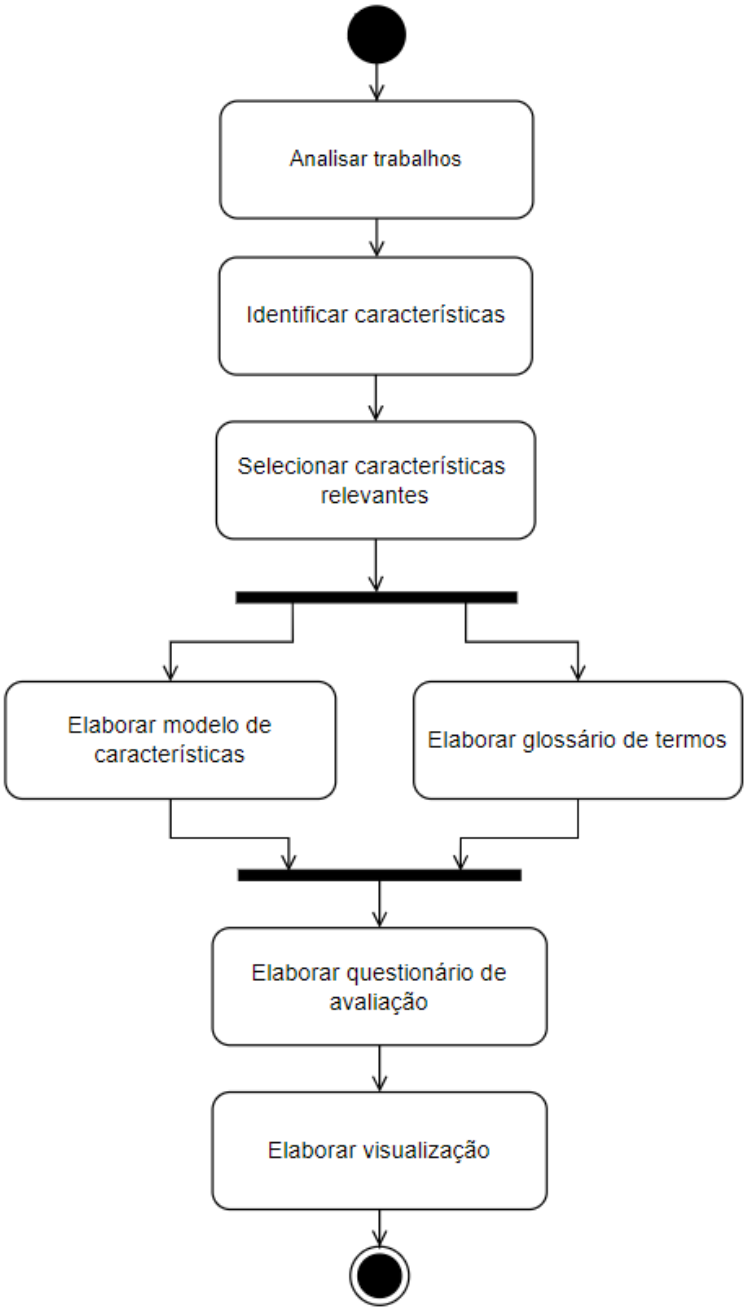
(iii) prover a **relação de interdependência** entre as características de acordo com a literatura; e

(iv) prover uma **ferramenta para analisar e avaliar** MS em relação a interesses não-funcionais, presentes no modelo de características proposto, de forma que seja possível expressar o grau de atendimento dessas características por meio de uma **estratégia de visualização de informação**.

3.2 Metodologia utilizada

A Figura 15 apresenta a metodologia utilizada no presente trabalho. A metodologia é composta pelas seguintes atividades:

Figura 15 - Metodologia utilizada.



Fonte: O autor, 2023.

- **Analisar trabalhos:** A análise dos trabalhos teve como objetivo explorar o contexto da AMS. Os trabalhos selecionados para análise são apresentados na Subseção 3.2.1;
- **Identificar características:** A análise textual dos trabalhos permitiu a identificação de características sobre a AMS. A lista de características é apresentada na Subseção 3.2.2;

- **Selecionar características relevantes:** O relacionamento dos trabalhos com as características identificadas possibilitou o entendimento do nível de importância de cada característica. Essa análise é apresentada na Tabela 7;
- **Elaborar modelo de características:** A elaboração do modelo de características teve como objetivo a representação do domínio de microsserviços, organizando as características de modo a permitir uma visão geral, sem detalhar a forma de geração de produtos desse domínio. O processo de modelagem não utilizou todas as etapas da engenharia de domínio. O modelo de características proposto encontra-se no Apêndice A;
- **Elaborar glossário de termos:** Um glossário de termos foi criado com o intuito de descrever um conjunto abrangente de conceitos presentes no modelo de características. Além disso, as relações de interdependência, de acordo com a literatura, estão presentes neste glossário, conforme o Apêndice B;
- **Elaborar questionário de avaliação:** Um questionário foi elaborado para a avaliação do atendimento às características da AMS presentes no modelo de características, conforme o Apêndice C; e
- **Elaborar visualização:** Uma técnica de visualização foi desenvolvida para expressar o grau de atendimento às características da AMS. Os fundamentos da técnica utilizada, assim como as informações relacionadas às versões desenvolvidas, são apresentados na Subseção 3.4.

3.2.1 Seleção dos trabalhos

Inicialmente, a proposta tinha como objetivo estudar a migração de sistemas legados para a AMS. Contudo, ao identificar a necessidade de organizar o conhecimento relacionado à AMS, a proposta foi adaptada para fornecer a análise e avaliação de MS, contribuindo para aumentar a conscientização das organizações sobre a complexidade na adoção dessa arquitetura.

Assim, de modo a ter maior diversidade de temas relacionados à AMS, foi feita a leitura, seleção e análise da literatura sobre microsserviços, visando identificar os aspectos centrais das AMS. A estratégia usada foi relacionar as características identificadas com os respectivos trabalhos para entender o nível de importância de cada característica, diante da amostra, por conveniência, com o objetivo de subsidiar a proposta inicial do modelo de características.

A Tabela 5 apresenta os trabalhos selecionados.

Tabela 5 - Trabalhos sobre a AMS selecionados após análise.

#	**Título**	**Referência**
T01	Effective migration of an automation system to microservice architecture	(LEHTOLA, 2020)
T02	From Monolithic Systems to Microservices - A Comparative Study of Performance	(TAPIA et al., 2020)
T03	Review of Methods for migrating software systems to microservices architecture	(STOJKOV; STOJANOV, 2021)
T04	Challenges and Solution Directions of Microservice Architectures - A Systematic Literature Review	(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)
T05	Construindo Aplicações Distribuídas com Microsserviços	(VILLAÇA; AZEVEDO; JR, 2018)
T06	A process to migrate legacy systems with business rules contained in stored procedures to a microservice-oriented architecture	(BARBOSA, 2020)
T07	Continuous software engineering - A microservices architecture perspective	(O'CONNOR; ELGER; CLARKE, 2017)
T08	Migrating Legacy Software to Microservices Architecture	(KAZANAVIČIUS; MAŽEIKA, 2019)
T09	O impacto do Uso de Micro Serviços na Evolução de uma Linha de Produto de Software	(BECKER, 2019)
T10	Does migrating a monolithic system to microservices decrease the technical debt?	(LENARDUZZI et al., 2020)
T11	Director - A cloud microservice selection framework	(COSTA, 2019)
T12	A Comparative Review of Microservices and Monolithic Architectures	(AL-DEBAGY; MARTINEK, 2018)
T13	A survey on microservices criticality attributes on established architectures	(SANTOS; WERNER, 2019)
T14	Uma abordagem de conformidade arquitetural para arquitetura de microsserviços	(ARAUJO, 2019)
T15	A Method to Detecting Artifact Anomalies in A Microservice Architecture	(FAHMI; HUANG; WANG, 2020)
T16	Microservices migration patterns	(BALALAIE et al., 2018)
T17	Sugestão de criticidade baseada em multicritério para arquiteturas estabelecidas orientadas a microsserviços	(SANTOS, 2020)
T18	From Monolithic Architecture to Microservices Architecture	(LAURETIS, 2019)
T19	From a Monolith to Microservices - The Effect of Multiview Clustering	(ASSELDONK, 2021)
T20	Extracting Microservices Candidates from Monolithic Applications: Interface Analysis and Evaluation Metrics Approach	(AL-DEBAGY; MARTINEK, 2020)

Fonte: O autor, 2023.

De acordo com a análise dos trabalhos selecionados sobre AMS, foi possível observar um grande interesse por abordagens de migração automáticas de AMO para a AMS, bem como relacionados à complexidade da arquitetura, considerando a coleta de um vasto número de características descrevendo essa arquitetura, sendo necessária a organização das mesmas.

3.2.2 Características identificadas

De forma a sintetizar os achados, foi feita uma organização das características identificadas. Tais características estão dispostas na Tabela 6.

Tabela 6 - Características identificadas sobre a AMS.

#	**Característica**	#	**Característica**
C01	Escalabilidade	C14	Testabilidade
C02	Tamanho	C15	Disponibilidade
C03	Autonomicidade	C16	Confiabilidade
C04	Comunicabilidade	C17	Componibilidade
C05	Implantabilidade	C18	Continuidade
C06	Manutenibilidade	C19	Orquestração
C07	Resiliência	C20	Monitorabilidade
C08	Acoplamento	C21	Coreografia
C09	Virtualização	C22	Elasticidade
C10	Coesão	C23	Balanceamento de carga
C11	Distributividade	C24	Reusabilidade
C12	Propriedade dos dados	C25	Segurança
C13	Desempenho	C26	Descoberta de serviço

Fonte: O autor, 2023.

Nessa identificação, observou-se a frequência elevada de características requeridas quando se utiliza tecnologias nativas de nuvem. Além disso, algumas características identificadas foram desconsideradas, seja por estarem relacionadas a técnicas de uma característica, já presente na listagem, como o uso de *Log* e a Monitorabilidade (C20); seja por estarem associadas a processos de desenvolvimento, como a Agilidade, que foi citada em muitos trabalhos.

A Tabela 7 apresenta o relacionamento dos trabalhos com as características identificadas, permitindo o entendimento do nível de importância de cada característica.

Com base nessa análise mais apurada, foi observado que a característica Escalabilidade (C1) esteve presente em todos os trabalhos analisados, enquanto três características – Reusabilidade (C24), Segurança (C25) e Descoberta de serviço (C26) – foram descritas em seis dos vinte trabalhos.

Tabela 7 - Análise do nível de importância de cada característica na amostra.

	T01	**T02**	**T03**	**T04**	**T05**	**T06**	**T07**	**T08**	**T09**	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**T19**	**T20**	**TOTAL**
C01	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	20
C02	⊕	⊕	⊕	⊕	⊕		⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	19
C03	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		19
C04	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		⊕	⊕	⊕	19
C05	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		19
C06	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		⊕	⊕	⊕	⊕		⊕		⊕	⊕	⊕	⊕	17
C07	⊕	⊕	⊕	⊕	⊕	⊕		⊕	⊕	⊕		⊕	⊕	⊕		⊕	⊕	⊕	⊕	⊕	17
C08	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		⊕		⊕	⊕			⊕	⊕	⊕	⊕	16
C09	⊕	⊕	⊕		⊕	⊕	⊕	⊕	⊕	⊕		⊕	⊕		⊕	⊕		⊕			14
C10	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕	⊕		⊕			⊕		⊕	⊕	13
C11	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕			⊕	⊕						13
C12	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕		⊕			⊕	⊕	⊕					13
C13	⊕	⊕	⊕	⊕	⊕			⊕		⊕		⊕	⊕		⊕		⊕	⊕			12
C14	⊕	⊕	⊕	⊕	⊕	⊕	⊕			⊕	⊕	⊕		⊕			⊕				12
C15		⊕	⊕	⊕	⊕	⊕			⊕		⊕	⊕				⊕	⊕		⊕		11
C16	⊕		⊕				⊕	⊕	⊕		⊕		⊕				⊕	⊕		⊕	10
C17	⊕	⊕		⊕	⊕		⊕			⊕					⊕	⊕		⊕	⊕		10
C18	⊕	⊕		⊕	⊕	⊕	⊕	⊕	⊕		⊕		⊕								10
C19	⊕	⊕	⊕	⊕		⊕	⊕							⊕	⊕	⊕					9
C20	⊕	⊕	⊕	⊕	⊕	⊕		⊕	⊕							⊕					9
C21	⊕	⊕			⊕	⊕				⊕		⊕		⊕	⊕						8
C22			⊕	⊕	⊕			⊕	⊕	⊕						⊕	⊕				8
C23				⊕	⊕	⊕	⊕						⊕	⊕		⊕					7
C24			⊕			⊕					⊕	⊕	⊕							⊕	6
C25	⊕		⊕	⊕						⊕	⊕		⊕								6
C26	⊕	⊕		⊕		⊕						⊕		⊕							6
TOTAL	22	21	21	21	21	19	17	16	16	16	16	15	15	15	13	14	13	12	11	9	

Legenda: ⊕ significa que uma característica (C) foi mencionada ou descrita em um trabalho (T).

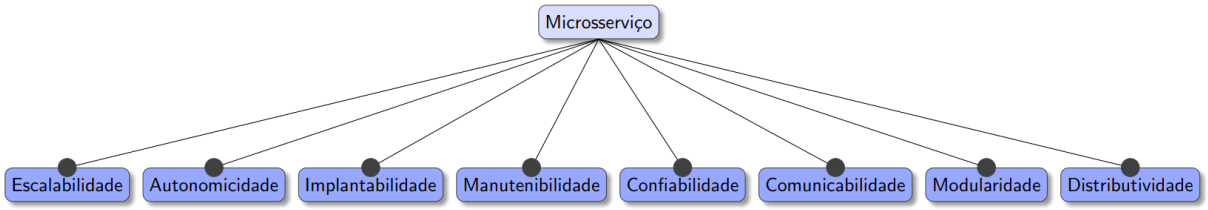
Fonte: O autor, 2023.

3.3 Modelo de características proposto

O processo de nomenclatura das características teve como objetivo definir termos que fossem desvinculados de detalhes de implementação, de nomes de tecnologias e de nomes de fornecedores. Além disso, quando possível, os termos foram submetidos às regras de derivação sufixal do português, com a adição do sufixo *-idade*, formando substantivos abstratos, que designam qualidade, modo de ser, estado e propriedade.

Um exemplo desse processo foi a substituição do termo Autonomia para Autonomicidade, referenciado no modelo de características proposto, conforme a Figura 16, que apresenta o primeiro nível desse modelo.

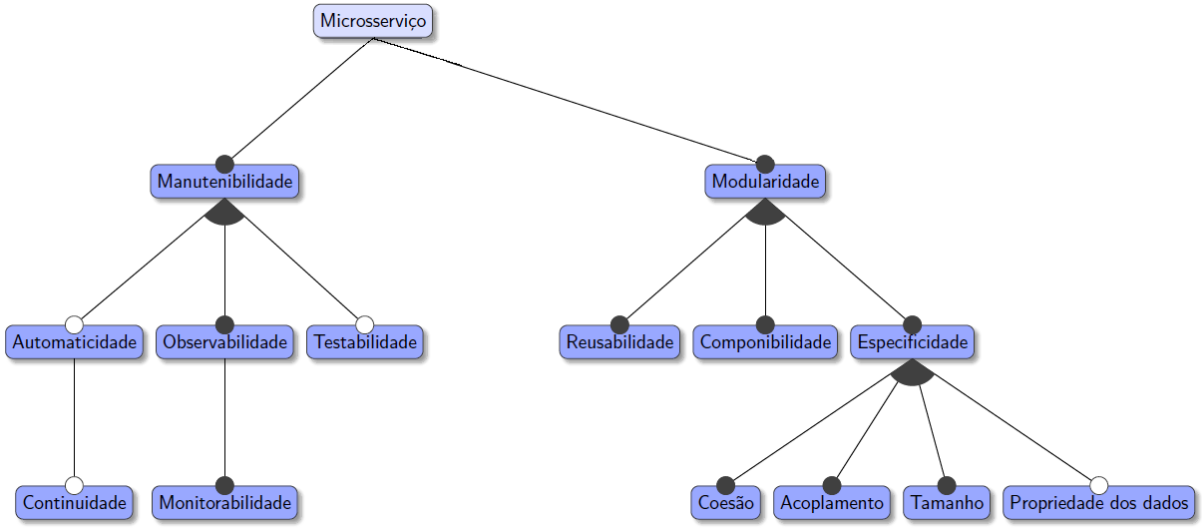
Figura 16 - Primeiro nível do modelo de características proposto.



Fonte: O autor, 2023.

Como pode ser visto na Tabela 6, 26 características foram selecionadas. Durante a modelagem de características foi necessária a criação de relacionamentos entre as características com a adição de características mais abstratas, visando agrupar características mais específicas, contribuindo com a compreensão e aderência ao domínio do modelo proposto. Devido a isso, o modelo de características proposto, apresentado em detalhes no Apêndice A, contém características adicionais, a saber: Automaticidade, Observabilidade, Modularidade e Especificidade. Essas características estão ilustradas na Figura 17.

Figura 17 - Características incluídas durante a modelagem de características.



Fonte: O autor, 2023

A Tabela 8 apresenta a classificação das características identificadas no presente estudo em relação aos aspectos organizacional, operacional e técnico, mencionados na introdução.

Tabela 8 - Classificação das características identificadas em relação aos aspectos organizacional, operacional e técnico.

Organizacionais	**Operacionais**	**Técnicos**
Automaticidade	Autonomicidade	Acoplamento
Continuidade	Balanceamento de carga	Coesão
	Componibilidade	Confiabilidade
	Comunicabilidade	Desempenho
	Coreografia	Disponibilidade
	Descoberta de serviço	Especificidade
	Distributividade	Manutenibilidade
	Elasticidade	Modularidade
	Escalabilidade	Resiliência
	Implantabilidade	Reusabilidade
	Monitorabilidade	Segurança
	Observabilidade	Tamanho
	Orquestração	Testabilidade
	Propriedade dos dados	
	Virtualização	

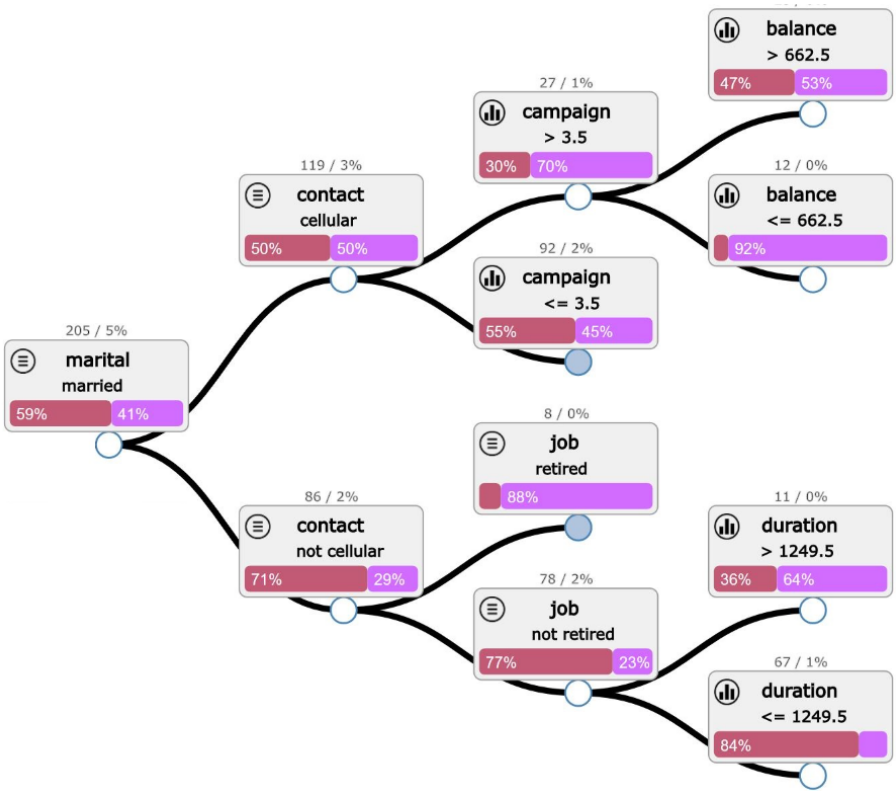
Fonte: O autor, 2023.

Conforme a Tabela 8, nos aspectos organizacionais, encontram-se interesses relacionados a processos ágeis. Os aspectos operacionais contemplam os interesses com forte relação à AMS. Por fim, nos aspectos técnicos, encontram-se interesses comuns, que normalmente estão presentes em qualquer arquitetura de *software*.

3.4 Técnica de Visualização de Dados

Em relação à estratégia de visualização de Dados para expressar o grau de atendimento às características da AMS, utiliza-se como fundamento a técnica de visualização análoga à utilizada na árvore em Fractalytics (2019), que visa facilitar e melhorar a legibilidade da árvore, conforme ilustrado na Figura 18.

Figura 18 - Técnica de visualização usando uma árvore.



Fonte: Adaptado de Fractalytics (2019)

A técnica apresentada na Figura 18 foi adaptada para suportar os requisitos relacionados ao presente estudo e está na segunda versão desde o início dos testes. A caracterização de ambas, assim como a motivação para cada escolha de cor, comportamento e funcionamento estão descritas a seguir:

Primeira versão

A primeira versão da técnica de visualização foi adaptada para ser utilizada com as informações do questionário de avaliação proposto, que, inicialmente, utilizava questões com respostas fechadas do tipo “sim”, “não” e “não se aplica”, em que cada usuário selecionava apenas uma dessas como resposta.

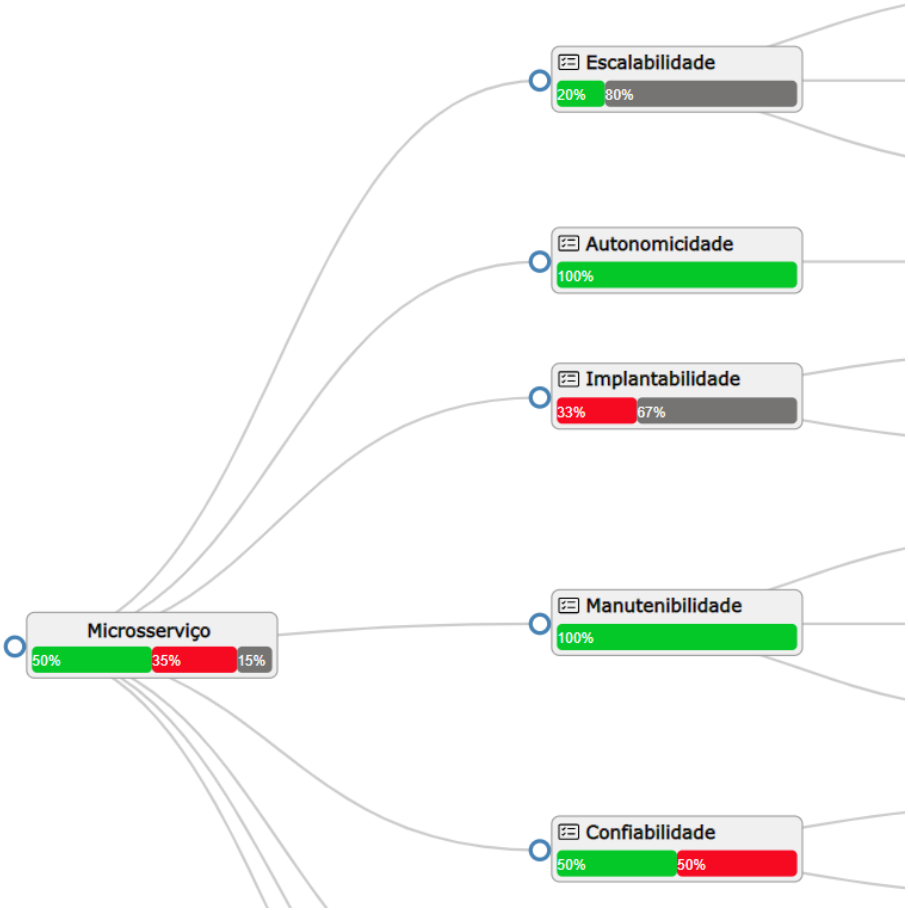
Desse modo, cada folha da árvore podia exibir até três barras horizontais nas cores verde, vermelho e cinza, respectivamente relacionadas às respostas “sim”, “não” e “não se aplica” do questionário. As cores foram escolhidas com o objetivo de criar uma visualização intuitiva para o usuário.

De acordo com Rost (2018), a utilização do verde e do vermelho permite um forte contraste, favorecendo a percepção da representação de opostos. O verde foi utilizado para a conformidade, enquanto o vermelho para a não conformidade ao atendimento às características de acordo com as perguntas do questionário de avaliação. A cor cinza normalmente é utilizada para representar elementos de contexto geral, anotações menos importantes ou elementos de uma *interface* não selecionados por usuários (ROST, 2018). Além disso, a cor cinza contribui para destacar outras cores utilizadas em conjunto. Devido a isso, ela foi utilizada para expressar quais perguntas do questionário de avaliação não se aplicam ao contexto da avaliação.

A raiz da árvore, a folha abstrata Microsserviço, apresentava os percentuais do somatório das respostas “sim”, “não” e “não se aplica” relacionados a todas as outras folhas. Por sua vez, cada folha da árvore, com exceção da raiz, apresentava os percentuais do somatório das respostas supracitadas, de acordo com cada característica.

A Figura 19 apresenta a primeira versão da técnica de visualização proposta para expressar o grau de atendimento às características da AMS.

Figura 19 - Primeira versão da técnica de visualização proposta.



Fonte: O autor, 2023

Segunda versão

Uma análise da técnica de visualização proposta foi realizada pelos pesquisadores. Nessa análise foi possível identificar algumas oportunidades para melhorias. Identificou-se que a utilização da cor vermelha para denotar erros é praticamente uma convenção, não sendo interessante o resultado de uma avaliação (*e.g.*, quando uma organização não apresenta determinado conhecimento) ser associado a algo negativo. Ainda, segundo Maia (2013), a utilização das cores vermelha e verde poderia impactar indivíduos portadores do daltonismo, que apresentam dificuldades na diferenciação dessas cores.

Outra questão a ser melhorada foi o resultado da avaliação das características. A avaliação não considerava o fato de existir perguntas mais relevantes do que outras. Além disso, a utilização dos percentuais das respostas “sim”, “não” e “não se aplica” para representar o resultado da avaliação de uma característica foi considerado demasiado e pouco efetivo.

Assim, após a identificação dos pontos de melhoria apresentados, uma segunda versão da técnica de visualização proposta foi realizada. Nessa versão, optou-se pela escolha de cores acessíveis e que apresentassem um bom contraste entre elas.

A avaliação também foi ajustada. Cada pergunta passou a ter um peso associado. O somatório dos pesos das perguntas de uma característica deve ser igual a um. Se a característica apresentar somente uma pergunta associada, seu peso será um. Além disso, as opções de respostas "sim", "não" e "não se aplica" foram substituídas por níveis de atendimento às características. Com isso, o usuário ao responder uma pergunta, deve optar por um dos seguintes níveis de atendimento:

- **SC:** Não há conhecimento sobre o assunto;
- **SI:** Sabe-se da importância, porém o assunto não foi priorizado;
- **AISP:** Tem-se a informação, o atendimento é insatisfatório, sem plano para resolução;
- **AICP:** Tem-se a informação, o atendimento é insatisfatório, com plano para resolução;
- **AS:** Tem-se a informação e o atendimento é satisfatório; e
- **ASCP:** Tem-se a informação, o atendimento é satisfatório, com plano de mitigação.

Além disso, um medidor (*gauge*) foi desenvolvido com o objetivo de facilitar a interpretação dos resultados da avaliação pelo usuário. Nesse medidor, os níveis de atendimento são representados por meio de uma escala sequencial de cores, que utiliza um gradiente classificado, variando de um tom claro ao escuro. O medidor apresenta as legendas: 0, 20, 40, 60, 80 e 100.

A avaliação de cada característica é realizada por meio do somatório do produto do valor da resposta de cada pergunta com o respectivo peso associado, resultando em um valor, entre 0 e 100, conforme a equação a seguir:

$$AV_{caracteristica} = \sum_{i=1}^{n} (x_i * w_i)$$

onde:

n = número de perguntas associadas a uma característica.

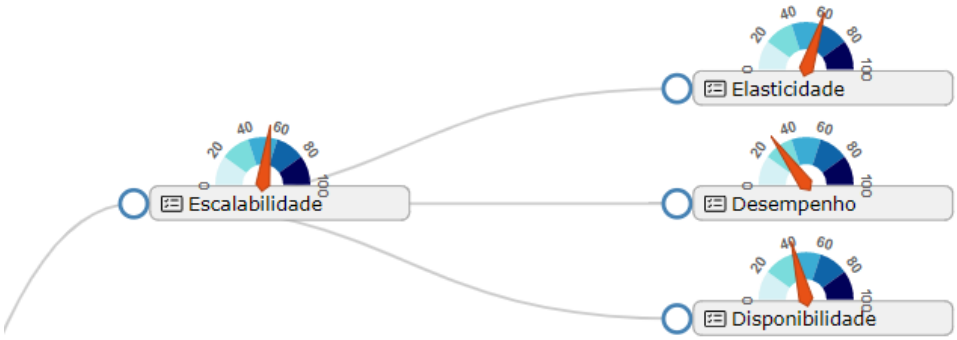
x = possíveis valores de resposta a uma pergunta: 0, 20, 40, 60, 80 e 100.

w = peso de uma pergunta. O somatório de todos os pesos das perguntas associadas a uma característica é 1.

O medidor apresenta um ponteiro indicador que se move de acordo com o resultado avaliado. Cada folha da árvore, com exceção da raiz, exibe o medidor.

Essas alterações foram realizadas para criar uma avaliação mais robusta e efetiva, bem como para motivar organizações que estejam no caminho da melhoria. A Figura 20 apresenta a segunda versão da técnica de visualização proposta para expressar o grau de atendimento às características da AMS.

Figura 20 - Segunda versão da técnica de visualização proposta.



Fonte: O autor, 2023.

3.4.1 O *framework* D3.js

O *framework* D3.js (*Data-Driven Documents*) é uma biblioteca JavaScript para a criação de visualizações de dados dinâmicas e interativas, que utiliza componentes de visualização úteis e a manipulação de elementos DOM (*Document Object Model*) (BOSTOCK; OGIEVETSKY; HEER, 2011).

O Projeto D3 teve início em 2011 e foi derivado do Projeto Protovi conduzido pela equipe de visualização da Universidade de Stanford em 2009 (LUO et al., 2020). Desde então, segundo Chi, Huang e Chuang (2016), o *framework* se tornou famoso e popular para a utilização de visualização de dados na *web*. Está disponível gratuitamente sob licença BSD (*Berkeley Software Distribution*) (LEE; JO; KIM, 2014).

O D3.js permite que visualizações sejam criadas diretamente em páginas HTML (*HyperText Markup Language*) por meio de gráficos vetorizados SVG (*Scalable Vector Graphics*) e de CSS (*Cascading Style Sheets*), com desempenho considerável e sem a dependência de navegadores específicos. SVG é um formato baseado em XML utilizado para descrever gráficos vetoriais que são independentes de resolução. Esse formato suporta diversos elementos gráficos tais como formas, caminhos, texto e efeitos de filtro (BAO; CHEN, 2014).

De acordo com Chi, Huang e Chuang (2016), o *framework* trabalha com diversos tipos de dados, que após serem carregados, permitem a interação de usuários por meio do navegador.

Os dados brutos são transformados em formatos de dados de entrada específicos, tais como CSV (*Comma-separated values*), JSON (*JavaScript Object Notation*), TSV (*Tab-separated values*) e XML (*Extensible Markup Language*). Na sequência, os dados de entrada são convertidos em um conjunto de parâmetros especificando os gráficos, que são renderizados em um arquivo SVG.

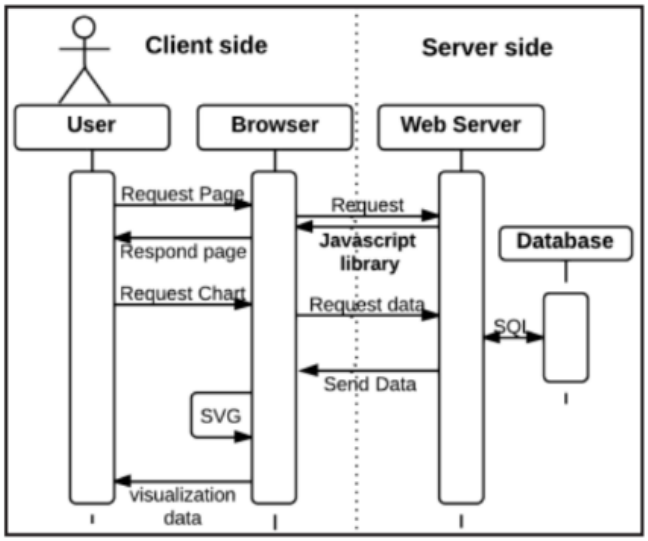
A Figura 21 apresenta o fluxo de transformação de dados no D3, desde o dado bruto até a visualização por meio do arquivo SVG.

Figura 21 - Fluxo de transformação de dados no D3.js.

Fonte: Adaptado de Lee, Jo e Kim (2014).

A configuração do D3 exige que a biblioteca seja referenciada no próprio código HTML. A Figura 22 apresenta um diagrama de sequência representando a resposta do servidor *web* com a biblioteca D3.

Figura 22 - Fluxo da visualização entre cliente e servidor no D3.js.



Fonte: Adaptado de Lee, Jo e Kim (2014).

O D3 oferece suporte a algoritmos avançados de visualização, como geografia, geometria e comportamentos. Além disso, possui diversas funções para auxiliar na visualização e análise de dados: ordenação, embaralhamento, permutação, junção e também

suporte a manipulação de algumas estruturas de dados, como *arrays*, *maps* e *collections* (LEE; JO; KIM, 2014).

3.5 **Considerações finais**

Neste capítulo, foram apresentados a metodologia, as atividades, a estratégia e os critérios utilizados para se atingir alguns dos objetivos deste trabalho. O capítulo a seguir apresenta a plataforma *web* desenvolvida para analisar e avaliar MS.

4 APOIO COMPUTACIONAL PARA A AVALIAÇÃO

Este capítulo descreve a plataforma *web* InA^{2}rMS (**In**strumento de apoio à **A**valiação da **Ar**quitetura de **M**icros**S**erviço) desenvolvida para a análise e avaliação de MS. Além deste objetivo, a plataforma apoia as atividades previstas no Capítulo 5, facilitando o alcance aos participantes, melhorando a celeridade dos processos de coleta e de análise de dados do experimento.

4.1 Introdução

A partir da definição da criação de um instrumento de apoio à avaliação pelos pesquisadores, surgiu a necessidade de gerenciar os participantes envolvidos, os artefatos sendo avaliados, as avaliações e também variáveis relacionadas a esse processo. O resultado da avaliação de um conjunto abrangente de características deve ser intuitivo e preciso para o participante. Além disso, a distância física entre pesquisadores e participantes não deve criar dificuldades, como a indisponibilidade de agenda.

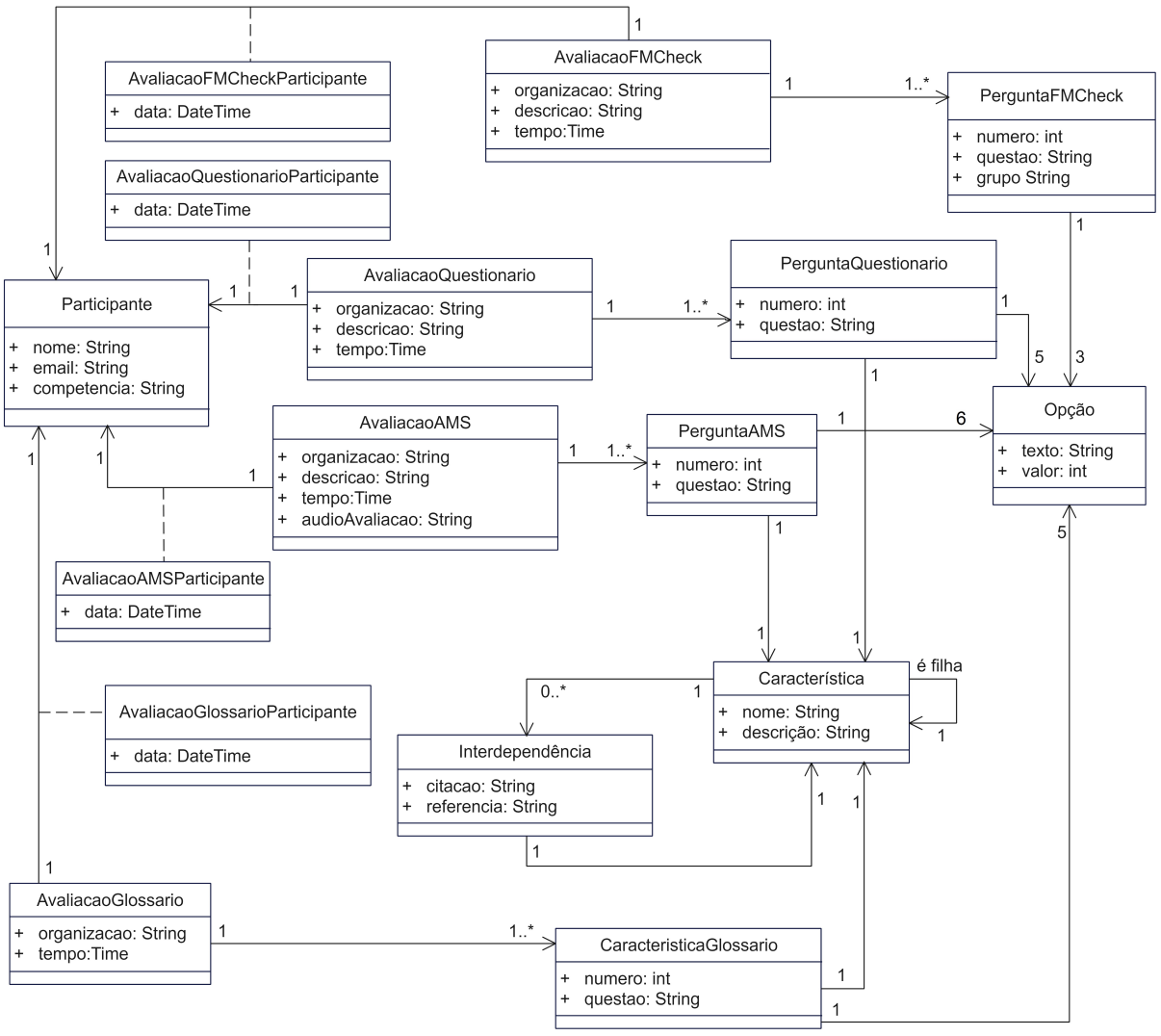
Esses requisitos, juntamente com os artefatos gerados pela aplicação da metodologia apresentada no Capítulo 3, como o modelo de características, o glossário de termos, o questionário de avaliação e a técnica de visualização, foram analisados e, após a modelagem do nível conceitual, foi possível descrever os tipos de objetos, propriedades, restrições e relacionamentos estáticos existentes, que subsidiaram a construção da plataforma InA^{2}rMS. A Figura 23 apresenta o diagrama das classes da plataforma InA^{2}rMS.

4.2 Arquitetura e implementação

A plataforma utiliza o Asp.Net MVC (*Model-View-Controller*) com o .Net *Framework* versão 4.6, na linguagem de programação C# e o SGBD MySql, versão 8.0.28.

O padrão MVC, conforme (GAMMA et al., 1994), possibilita a divisão da aplicação em três camadas bem definidas: o *Model*, que é responsável por gerenciar o comportamento dos dados, por meio de funções, lógicas e regras; a *View*, que é responsável por apresentar as informações para o usuário; e o *Controller*, que é responsável por receber as requisições dos usuários. A utilização do padrão MVC traz como benefício principal o isolamento das regras de negócios da lógica de apresentação. A plataforma utiliza dois *frameworks* de código-aberto na camada *View*: o Bootstrap, na versão 4.1.1, que é utilizado para o desenvolvimento *web*, com a reutilização de componentes baseados em HTML e CSS; e o D3.js, na versão 5, introduzido no Capítulo 3, que é utilizado na visualização do resultado

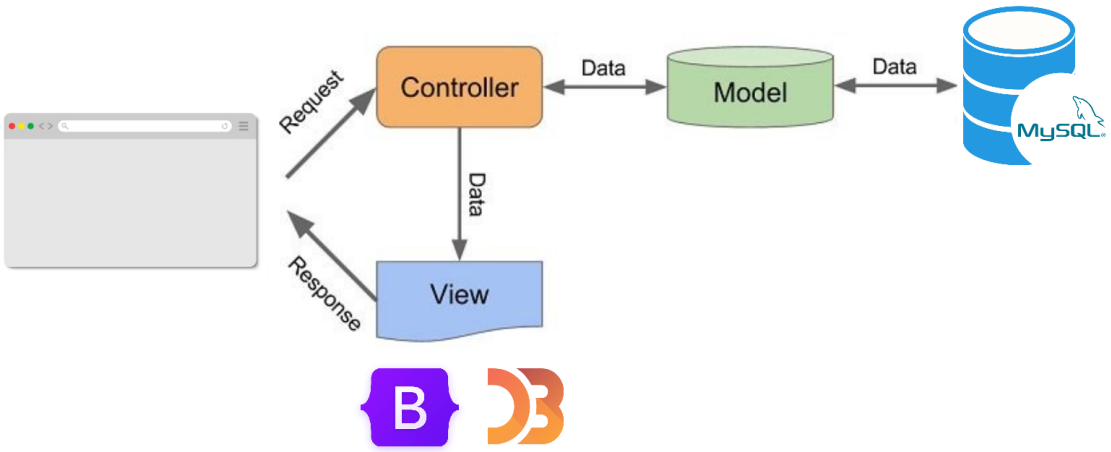
Figura 23 - Diagrama de classes da plataforma InA^{2}rMS.



Fonte: O autor, 2023.

da avaliação. A Figura 24 apresenta a arquitetura da plataforma.

Figura 24 - Arquitetura da plataforma InA^{2}rMS e tecnologias utilizadas.



Fonte: O autor, 2023.

4.3 **Principais visões da plataforma**

A plataforma prevê a participação de especialistas, que são convidados pelos pesquisadores conforme o perfil necessário para a avaliação de cada artefato presente nesse estudo. O participante, após ser cadastrado, recebe o convite contendo o endereço da plataforma (*https://www.ina2rms.com.br*) e também instruções para auxiliar a avaliação. A Figura 25 apresenta a visão inicial da plataforma. Acionando a opção participar, o especialista visualiza as avaliações disponíveis, conforme Figura 26.

Figura 25 - Visão inicial da plataforma.

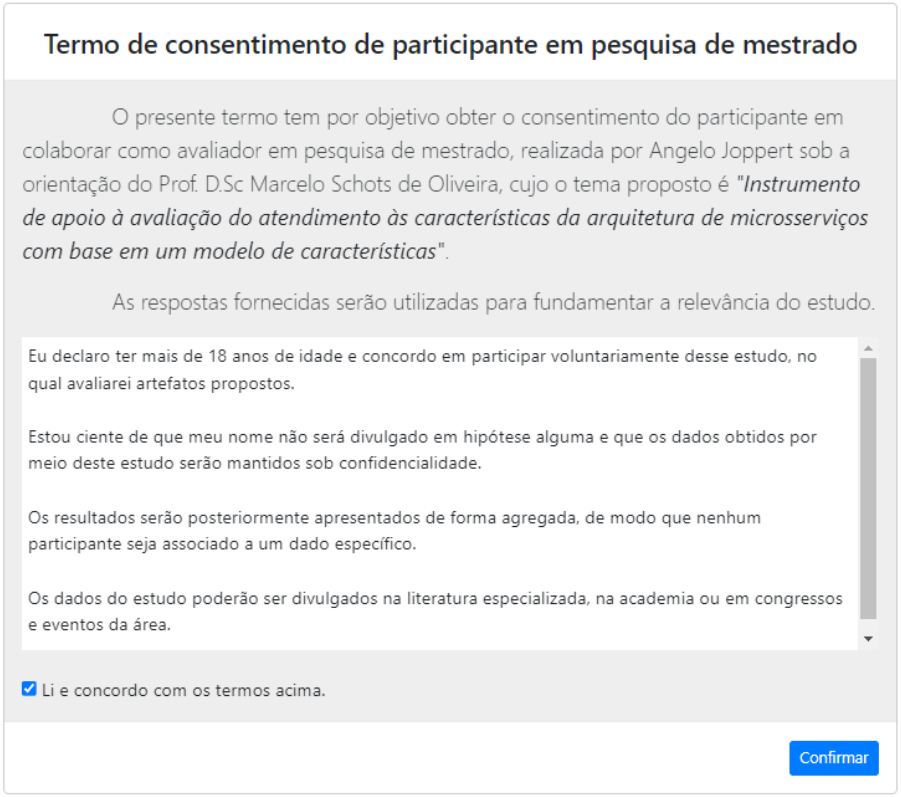
Fonte: O autor, 2023.

Figura 26 - Avaliações cadastradas para um participante.

Fonte: O autor, 2023.

Para iniciar uma avaliação, o participante deve concordar com o termo de consentimento que lhe é apresentado, conforme Figura 27.

Figura 27 - Termo de consentimento de participante em pesquisa de mestrado.



Fonte: O autor, 2023.

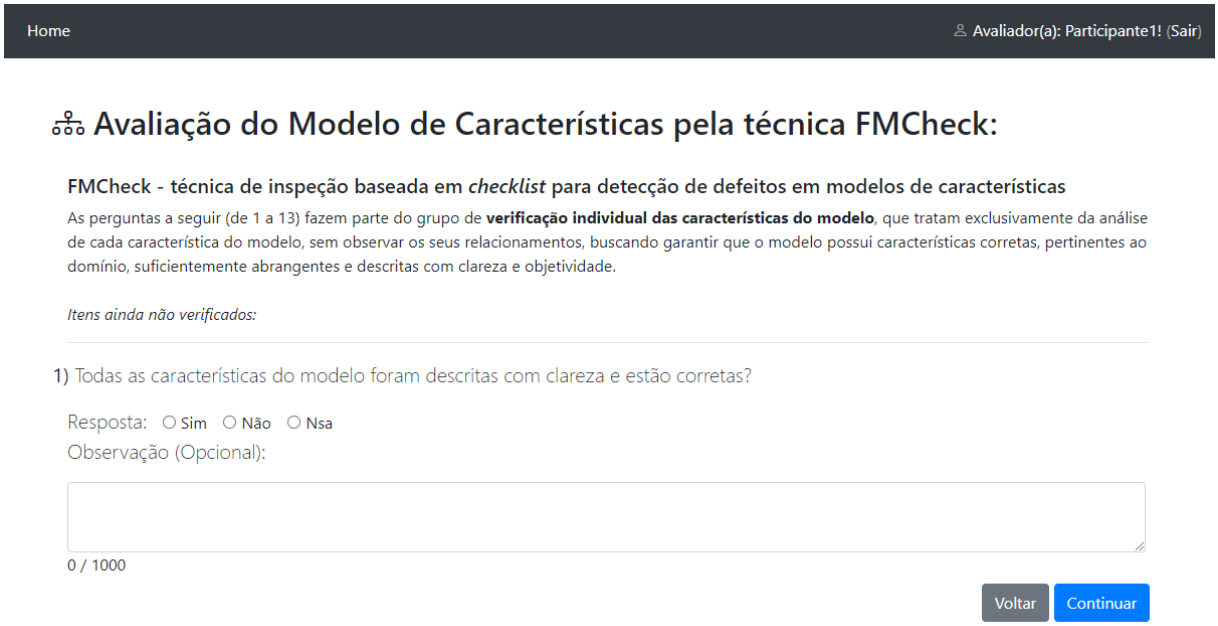
Avaliações

A plataforma prevê a avaliação dos seguintes artefatos:

Modelo de características

A plataforma incorporou o FMCheck, uma técnica de inspeção baseada em *checklist*, disponível no Anexo, para os participantes avaliarem o modelo de característica proposto. Com o objetivo de coletar os *feedbacks* relacionados a cada pergunta do *checklist*, foram disponibilizados campos do tipo caixa de texto. A Figura 28 apresenta a avaliação de um item do *checklist* por meio da plataforma.

Figura 28 - Incorporação do FMCheck à plataforma.



Fonte: O autor, 2023.

Glossário de termos

O glossário de termos, disponível no Apêndice B, é composto pela definição das características e também pelo relacionamento de interdependência entre as características. Cada item do glossário é avaliado quanto ao grau de pertinência da característica associada. Este artefato também foi incorporado à plataforma, conforme Figura 29.

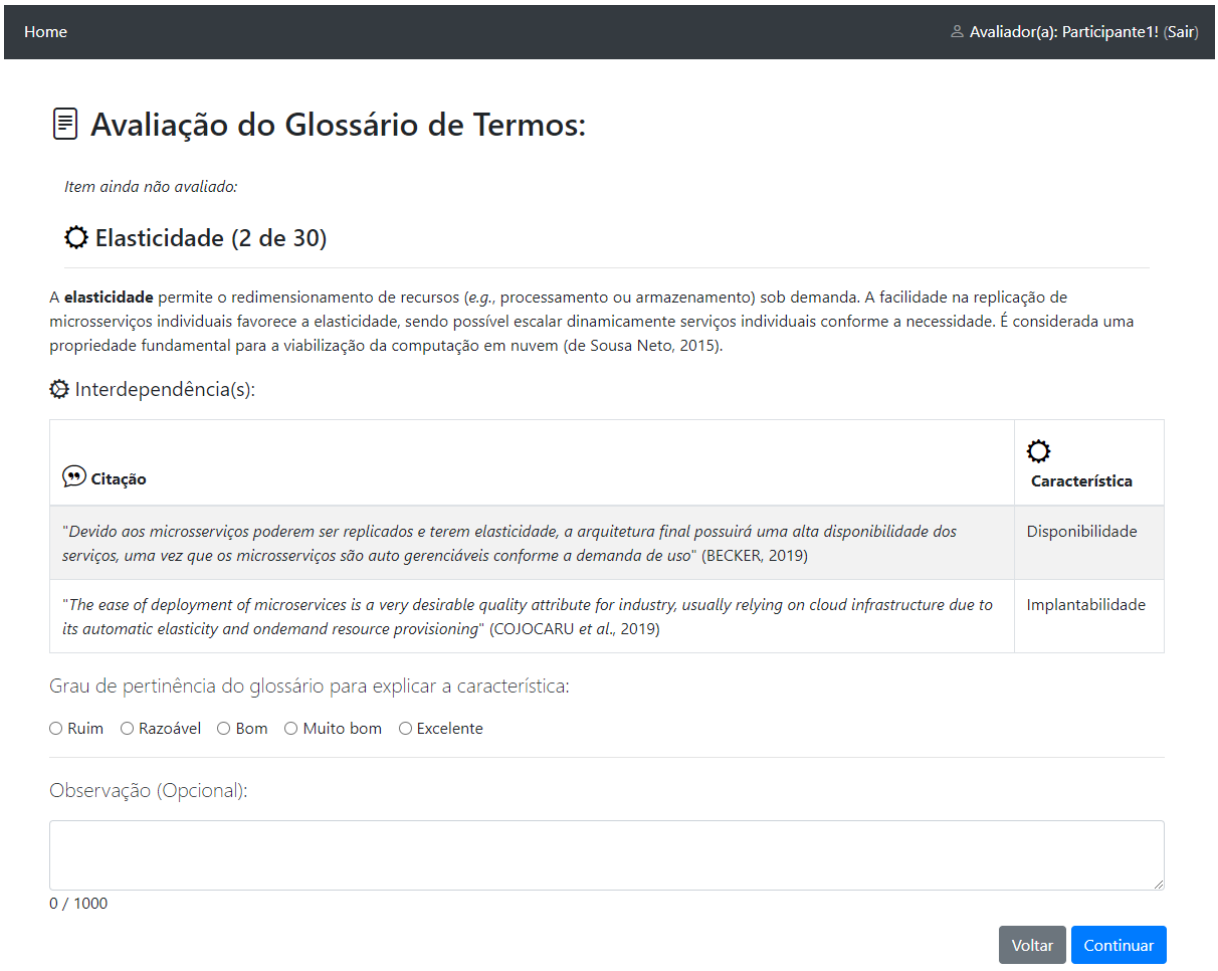
Questionário de avaliação

As perguntas formuladas para o questionário de avaliação, disponíveis no Apêndice C, são avaliadas quanto ao grau de pertinência para avaliar a característica associada. A Figura 30 apresenta a avaliação de uma das perguntas da característica Escalabilidade.

Aplicação do questionário de avaliação

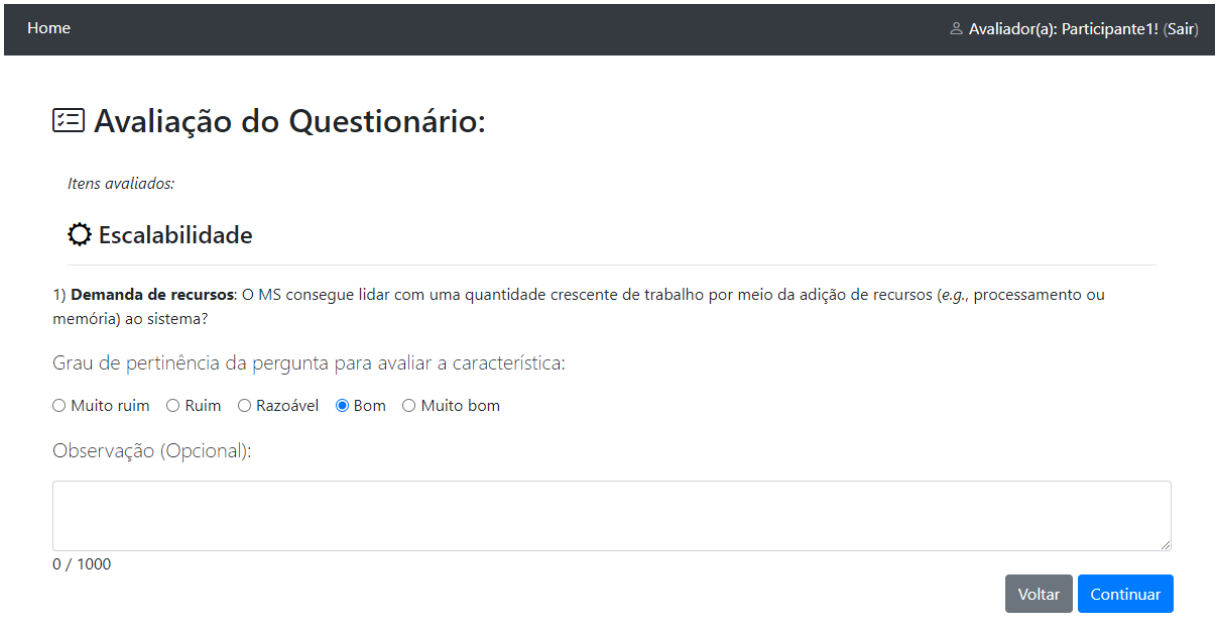
O questionário é aplicado em um MS selecionado pelo participante. Cada pergunta possui um peso associado, definido pelos pesquisadores. Na avaliação, de forma a evitar vieses, o participante não possui ciência dos pesos das perguntas. O participante deve selecionar apenas uma resposta entre os níveis de atendimento. A Figura 31 apresenta a aplicação do questionário na avaliação da característica Manutenibilidade.

Figura 29 - Avaliação de item do glossário de termos.



Fonte: O autor, 2023.

Figura 30 - Avaliação de perguntas do questionário.



Fonte: O autor, 2023.

Figura 31 - Aplicação do questionário na avaliação da característica Manutenibilidade.

Fonte: O autor, 2023.

A exibição do resultado da avaliação torna-se disponível para os participantes que concluíram a aplicação do questionário de avaliação. A Figura 32 apresenta a tela na qual os participantes acionam o resultado da visualização que utiliza a técnica de visualização.

Figura 32 - Aplicação do questionário de avaliação concluída.

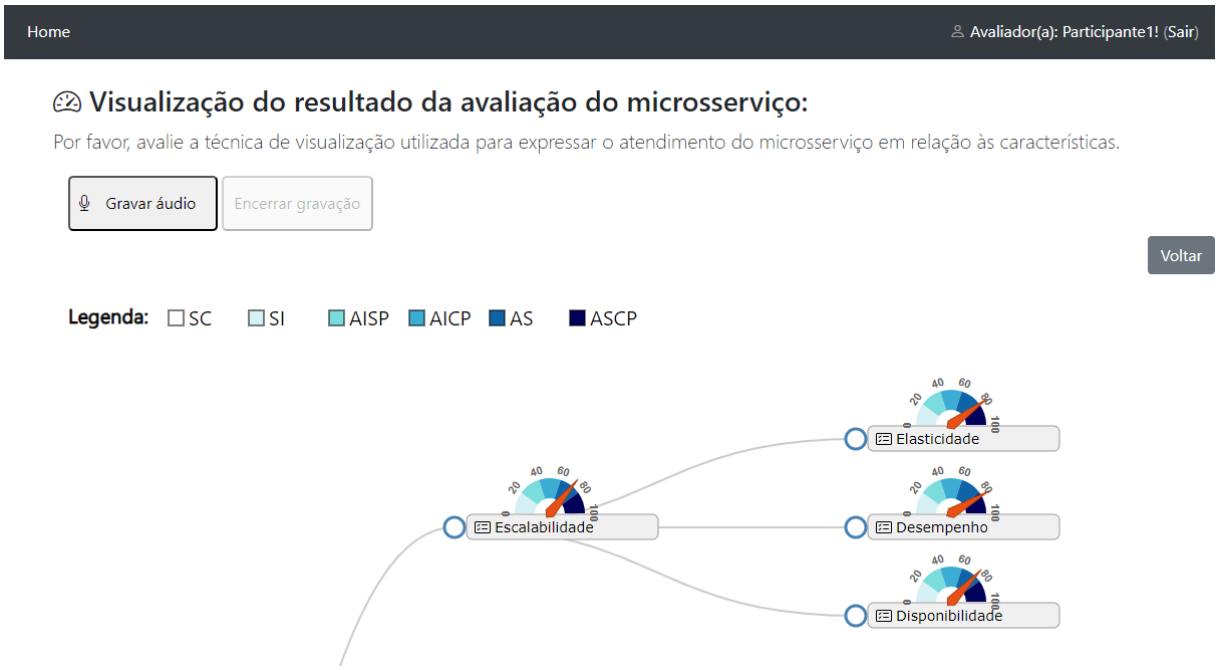
Fonte: O autor, 2023.

Técnica de visualização

A técnica de visualização, introduzida no Capítulo 3, só pode ser avaliada após a finalização da aplicação do questionário. Ao visualizarem a técnica proposta para expressar o grau de atendimento às características da AMS, os participantes gravam os áudios de suas avaliações por meio da *interface* disponibilizada, conforme as Figuras 33 e 34.

Algumas funcionalidades foram adicionadas à visualização proposta. Ao selecionar o ícone do questionário de avaliação, que está à esquerda do nome da característica nas folhas da árvore, é exibida uma visão com a(s) resposta(s) e respectivo(s) peso(s) relacionado(s) à característica, conforme a Figura 35.

Figura 33 - Exibição do resultado de uma avaliação com a opção da gravação do áudio do participante.

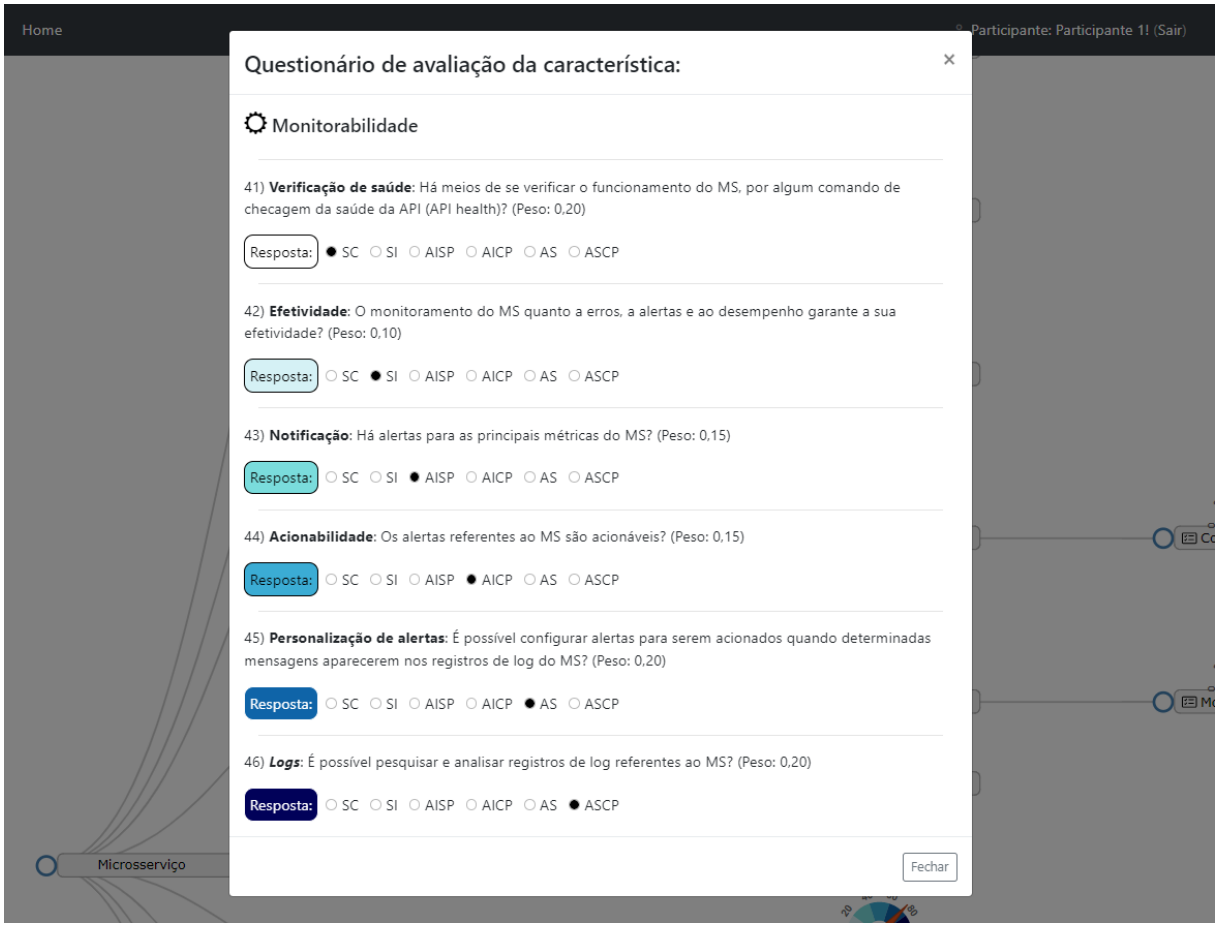


Fonte: O autor, 2023.

Figura 34 - Avaliação da técnica de visualização por participante com a gravação de áudio.

Fonte: O autor, 2023.

Figura 35 - Visão com respostas do questionário de avaliação da característica Monitorabilidade.



Fonte: O autor, 2023.

4.4 **Considerações finais**

Este capítulo apresentou a plataforma *web* InA^{2}rMS desenvolvida para a análise e avaliação de MS e também para apoiar o estudo experimental que é apresentado no capítulo a seguir. Este estudo possui o propósito de verificar sua aplicabilidade e o grau de concordância do ponto de vista de pesquisadores e profissionais no contexto de engenharia de *software*.

5 AVALIAÇÃO

Este capítulo descreve o experimento e suas etapas. O planejamento consiste no detalhamento dos artefatos, dos perfis dos participantes, dos cenários de avaliação, do tamanho das amostras e das métricas. A execução detalha as amostras e os dados coletados. A discussão dos resultados permite a análise dos dados obtidos. As ameaças à validade são apresentadas e, por fim, é realizada a conclusão do capítulo.

5.1 Planejamento

As avaliações dos artefatos propostos nesta abordagem estão divididas em três etapas. Na primeira, são avaliados o modelo de características e o glossário de termos. Na segunda, o questionário de avaliação. Na terceira etapa, aplica-se o questionário de avaliação em um microsserviço, bem como se avalia a técnica de visualização por meio do protocolo *think-aloud*, que é utilizado para registrar o *feedback* dos participantes ao interagirem com a visualização. Segundo Ericsson e Simon (1993), nessa técnica o participante é incentivado a falar tudo o que está pensando ao interagir com a *interface*. O gênero dos participantes não foi especificado.

Tamanho das amostras

O tamanho das amostras foi definido de acordo com critérios apontados por Merino et al. (2018) e (ROMNEY; WELLER; BATCHELDER, 1986 apud GUEST; BUNCE; JOHNSON, 2006).

A maioria das abordagens avaliando visualizações de informação envolve de um a cinco participantes (MERINO et al., 2018). Já conforme Romney, Weller e Batchelder (1986 apud GUEST; BUNCE; JOHNSON, 2006), amostras tão pequenas com quatro participantes podem fornecer informações extremamente precisas e com um bom nível de confiança, desde que os indivíduos possuam um alto grau de competência em relação ao objeto de pesquisa em questão.

Além disso, a definição dos tamanhos das amostras também levou em consideração questões relacionadas à disponibilidade dos participantes e à dificuldade em encontrar perfis específicos para a avaliação de determinados artefatos, como o modelo de características.

Os tamanhos das amostras, definidos nas subseções a seguir, levou em consideração as informações e dificuldades supracitadas.

Primeira avaliação

Como o modelo de característica possui cada elemento descrito pelo glossário de termos, estabeleceu-se que esses dois artefatos são avaliados pelo mesmo participante.

5.1.1 Modelo de características

Os participantes avaliam o modelo de características de acordo com a técnica de inspeção baseada em *checklist* FMCheck, apresentada no Anexo.

A técnica é composta por três grupos de verificação.

O terceiro grupo de verificação, que se refere à *verificação das regras de composição do modelo*, não foi aplicado, pois não foram utilizadas regras de composição para a criação do modelo de características proposto e também para não tornar a avaliação exaustiva. Dessa forma, a avaliação utiliza os seguintes grupos de verificação:

1. *Verificação individual das características do modelo*: tratam exclusivamente da análise de cada característica do modelo, sem observar os seus relacionamentos, buscando garantir que o modelo possui características corretas, pertinentes ao domínio, suficientemente abrangentes e descritas com clareza e objetividade; e

2. *Verificação dos relacionamentos entre as características do modelo*: orientam o inspetor a examinar os relacionamentos entre as características, analisando o quanto os relacionamentos entre as características do modelo o tornam compreensível, aderente ao domínio e implementável.

- **Perfil dos participantes:** Participantes com conhecimento em arquiteturas de *software* e em análise de domínio.

- **Cenário de avaliação:** É esperado que o participante realize a avaliação do modelo de características com o auxílio do glossário de termos.

- **Treinamento:** As informações necessárias para subsidiar a avaliação pelo participante estão disponíveis na plataforma *web* apresentada no Capítulo 4.

- **Tamanho da amostra:** Espera-se um número reduzido de participantes, pois se trata de um perfil muito restrito.

- **Métricas sendo avaliadas:** Tempo de conclusão da avaliação em segundos; lista de observações sobre cada pergunta do *checklist* e lista de observações gerais sobre a avaliação.

5.1.2 Glossário de termos

O glossário de termos é composto pela definição das características e também pelo relacionamento de interdependência entre as características, ambos baseados em referências da literatura. Cada item do glossário é avaliado quanto a sua pertinência para explicar a característica associada.

- **Perfil dos participantes:** Participantes com conhecimento em arquiteturas de *software* e em análise de domínio.
- **Cenário de avaliação:** É esperado que os participantes realizem a avaliação do glossário de termos com o auxílio do modelo de características.
- **Treinamento:** As informações necessárias para subsidiar a avaliação pelo participante estão disponíveis na plataforma *web* apresentada no Capítulo 4.
- **Tamanho da amostra:** Espera-se um número reduzido de participantes, pois se trata de um perfil muito restrito.
- **Métricas sendo avaliadas:** Tempo de conclusão da avaliação em segundos; grau de pertinência do glossário para explicar a característica usando escala *Likert* de cinco pontos (*e.g.*, "1-Muito ruim", "2-Ruim", "3-Razoável", "4-Bom", "5-Muito bom"), lista de observações sobre cada item do glossário e lista de observações gerais sobre a avaliação.

Segunda avaliação

5.1.3 Questionário de avaliação

As perguntas formuladas para o questionário são avaliadas quanto à pertinência para avaliar uma característica. O questionário é composto por oitenta e sete perguntas, agrupadas por características. O questionário foi elaborado levando em consideração o modelo de caraterísticas e as definições do glossário de termos.

- **Perfil dos participantes:** Participantes com conhecimento em arquiteturas de *software*.
- **Cenário de avaliação:** É esperado que o participante realize a avaliação do questionário com o auxílio do modelo de características e do glossário de termos.
- **Treinamento:** As informações necessárias para subsidiar a avaliação pelo participante estão disponíveis na plataforma *web* apresentada no Capítulo 4.

- **Tamanho da amostra:** Espera-se um número reduzido de participantes, pois se trata de um perfil com nível de conhecimento muito específico.
- **Métricas sendo avaliadas:** Tempo de conclusão da avaliação em segundos; grau de pertinência da pergunta para avaliar a característica usando escala *Likert* de cinco pontos (*e.g.*, "1-Muito ruim", "2-Ruim", "3-Razoável", "4-Bom", "5-Muito bom"), lista de observações sobre cada pergunta do questionário e lista de observações gerais sobre a avaliação.

Terceira avaliação

5.1.4 Técnica de visualização

Os participantes, após a aplicação do questionário de avaliação a um microsserviço selecionado, analisam a técnica de visualização proposta por meio do protocolo verbal *think-aloud*. Nesse método, o participante é instruído a manifestar em voz alta seus pensamentos sem interpretá-los, de forma simultânea e não-estruturada, durante a execução de uma determinada tarefa (ERICSSON; SIMON, 1993). Esse método é utilizado para avaliar documentos instrucionais, *websites* e *interfaces* (HAAK; JONG, 2003). Esse protocolo costuma identificar mais problemas de usabilidade do que entrevistas, questionários e análise de registros de *log* (HENDERSON et al., 1995 apud VELSEN; GEEST; KLAASSEN, 2007).

- **Perfil dos participantes:** Arquitetos de *software* ou desenvolvedores com experiência no desenvolvimento de microsserviços que avaliaram o microsserviço por meio da aplicação do questionário de avaliação.
- **Cenário de avaliação:** O participante deve selecionar um microsserviço para a aplicação do questionário de avaliação. Deve ser utilizado um monitor com resolução mínima de 1280x720 *pixels*. O microfone do computador deve estar funcionando para que o áudio da avaliação possa ser gravado.
- **Treinamento:** As informações necessárias para subsidiar a avaliação pelo participante estão disponíveis na plataforma *web* apresentada no Capítulo 4.
- **Atividades a serem realizadas na avaliação:** A técnica de visualização deve ser explorada pelos participantes por meio do acionamento das funcionalidades presentes na visualização. Além disso, deve ser gravado um áudio contendo sua avaliação em tempo real sobre a visualização.

- **Tamanho da amostra:** É esperada a participação de um a cinco participantes, conforme (MERINO et al., 2018).
- **Métricas sendo avaliadas:** Duração da avaliação da técnica de visualização em segundos e áudio da avaliação do participante.

5.2 Execução

O experimento teve início em 24 de abril de 2023 e se encerrou em 08 de junho de 2023, com a duração de 45 dias. Foi realizada uma amostragem não probabilística por conveniência. Os participantes foram convidados a colaborarem com o experimento, conforme o perfil necessário para a avaliação de cada artefato proposto. Não foram enviados convites abertos a fóruns muito amplos por não haver garantias da formação nos tópicos relacionados às etapas do experimento. A coleta de dados foi realizada por meio da plataforma *web* InA^{2}rMS, apresentada no Capítulo 4.

A Tabela 9 apresenta os participantes, suas profissões, formações acadêmicas e habilidades.

Tabela 9 - Formação acadêmica dos participantes de acordo com os artefatos a serem avaliados.

Participante	**Profissão**	**Formação acadêmica**	**Habilidades**
1	Engenheiro de *Software* Sênior	Mestrando	- Arquitetura de *software* - Arquiteturas de microserviços - Desenvolvimento em nuvem - Engenharia de *software* - Integração contínua/Entrega contínua - SOA (Arquitetura orientada a serviços) - Teste de *software* - Versionamento de *software*
2	Pesquisador	Doutor	- Arquitetura corporativa - Desenvolvimento Baseado em Componentes - Engenharia de qualidade de *software* - Gerência de configuração de *software*
3	Engenheiro de *Software* Sênior	Mestrando	- Aprendizado de máquina - Arquitetura de aplicações - DevOps - Microsserviços - Nuvem

continuação na próxima página

4	Pesquisador	Doutor	- Colaboração na engenharia de sistemas - Engenharia de *software* baseada em busca - Engenharia orientada a modelos - Gerenciamento de variabilidade - Modernização de *software* - Qualidade de *software* - Teste de *software*
5	Desenvolvedor *Web* Sênior	Bacharel em Informática	- Arquitetura de *software* - Desenvolvimento *Web*
6	Arquiteto de *Software* Sênior	Bacharel em Informática	- Agilidade - Arquitetura corporativa - Nuvem - Qualidade de *software*
7	Arquiteto de *Software* Sênior	Bacharel em Sistemas da Informação	- AWS - Devops - Microsserviços - Qualidade de *software*
8	Analista de Sistemas Sênior	Doutorando	- Conteinerização - Desenvolvimento *web* - *Mainframes* - Microsserviços
9	Analista de Sistemas Sênior	Doutorando	- Conteinerização - Inteligência artificial - Microsserviços
10	Analista de Sistemas Sênior	Mestrando	- Métricas de *software* - Modelagem de processos de negócios - Processos de *software*
11	Analista de Sistemas	Bacharel em Ciência da Computação	- Integração de serviços - Microsserviços - *Web* API
12	Arquiteto de *Software*	Bacharel em Sistemas da Informação	- API - Computação em nuvem - DevOps - Microsserviços
13	Engenheiro de *Software*	Bacharel em Engenharia Eletrônica	- API - Arquitetura de *software* - Microsserviços
14	Engenheiro de *Software*	Bacharel em Engenharia Eletrônica	- Arquitetura de *software* - Engenharia de requisitos - Teste de *software*
15	Engenheiro de *Software* Sênior	Bacharel em Tecnologia da Informação	- Azure - Computação em nuvem - Microsserviços - Restful API

continuação na próxima página

16	Arquiteto de *Software* Sênior	Bacharel em Ciência da Computação	- API - Arquitetura de *software* - DevSecOps - Entrega contínua - Microsserviços - Qualidade de código - Segurança de aplicações - Web services
17	Engenheiro de *Software*	Doutor	- Aplicações corporativas - Arquitetura de *software* - Desenvolvimento orientado a comportamento - Microsserviços
18	Líder técnico	Bacharel em Ciência da Computação	- Azure - Computação em nuvem - Conteinerização - Entrega contínua - Microsserviços - Qualidade de código
19	Arquiteto de *Software* Sênior	Tecnólogo em Redes de Computadores	- Arquitetura de *software* - Microsserviços - Qualidade de código - REST/RESTful APIs - TDD (*Test-Driven Development*)
20	Arquiteto de *Software* Sênior	Bacharel em Ciência da Computação	- Arquitetura de *software* - DevSecOps - Microsserviços - OWASP (*Open Web Application Security Project*) - RESTful API - *Web* API - *Webservices*
21	Arquiteto de *Software* Sênior	Ensino Médio	- Arquiteto de soluções - Arquiteturas RESTFul - AWS - Azure - Computação em nuvem - Conteinerização - *Webservices*

Fonte: O autor, 2023.

5.2.1 Modelo de características

A amostra foi formada por mestrandos e doutores, selecionados por experiência prévia em trabalhos acadêmicos relacionados ao tema e pela indicação realizada por professores. Foi obtida uma amostra com 4 participantes, resultando em uma taxa de resposta

correspondente a 23,53%.

O instrumento de pesquisa relacionado ao modelo de características foi composto por 29 perguntas fechadas do *checklist* FMCheck. Os participantes, de forma opcional, puderam registrar observações específicas e também gerais para cada pergunta na avaliação do artefato. Os dados coletados na avaliação do modelo de características são apresentados no Apêndice D.

5.2.2 Glossário de termos

A amostra é a mesma do modelo de características. O instrumento de pesquisa relacionado ao glossário de termos foi composto por 30 perguntas fechadas, em que o participante avaliou o grau de pertinência de cada item para explicar a característica associada. Os participantes, de forma opcional, puderam registrar observações específicas e também gerais para cada pergunta na avaliação do artefato. Os dados coletados na avaliação do glossário de termos são apresentados no Apêndice E.

5.2.3 Questionário de avaliação

A amostra foi formada por especialistas com conhecimento em arquiteturas de *software* (desenvolvedores e arquitetos de *software*). Alguns foram selecionados conforme o perfil disponível na plataforma Linkedin, uma rede social focada em negócios e emprego. Foi obtida uma amostra com 18 participantes, dos quais 3 foram eliminados por não terem concluído a avaliação do questionário e outros 4 por desistência. A amostra final, sem apresentar dados ausentes e sem contar com desistências, foi composta por 11 participantes. Assim, essa avaliação apresenta as seguintes taxas: de resposta igual a 52,38%, de desistência igual a 19,05% e de não concluintes igual a 14,29%.

O instrumento de pesquisa relacionado ao questionário de avaliação foi composto por 87 perguntas fechadas agrupadas em características, no qual os participantes avaliaram o grau de pertinência da pergunta para avaliar a característica. Os participantes, de forma opcional, puderam registrar observações específicas e também gerais para cada pergunta na avaliação do artefato. Os dados coletados são apresentados no Apêndice F.

5.2.4 Técnica de visualização

A amostra foi formada por especialistas (arquitetos de *software* ou desenvolvedores) com experiência no desenvolvimento de microsserviços. Alguns participantes foram

selecionados por experiência prévia em trabalhos acadêmicos e, conforme o perfil disponível na plataforma Linkedin. Foi obtida uma amostra com 11 participantes, da qual um participante foi eliminado por não ter concluído a avaliação da técnica de visualização por meio do protocolo *think-aloud* e outros 4 por desistência, tendo um deles informado que não se sentia confortável em enviar dados de voz. A amostra final, sem apresentar dados ausentes e sem contar com desistências, foi composta por 6 participantes. Assim, essa avaliação apresenta as seguintes taxas: de resposta igual a 31,58%, de desistência igual a 21,05% e de não concluintes igual a 5,26%.

O questionário de avaliação foi aplicado em um MS selecionado pelos participantes, que acionaram a técnica de visualização e a avaliaram por meio do protocolo *think-aloud*. Os dados coletados constam no Apêndice G.

5.3 **Discussão dos resultados**

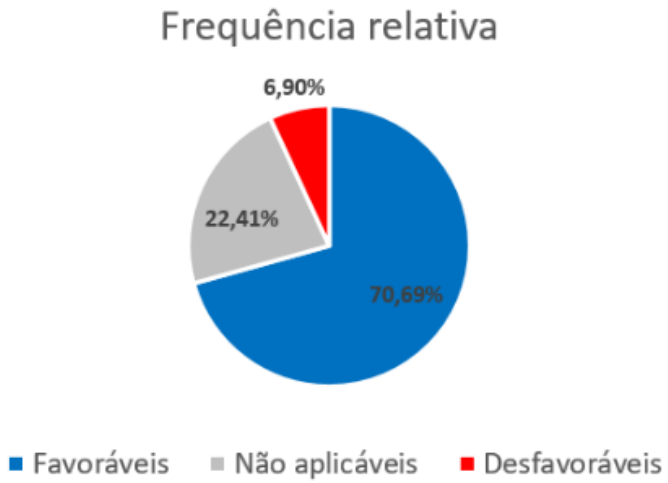
A seguir, são apresentados os resultados das avaliações de acordo com cada artefato.

5.3.1 Modelo de características

O tempo médio da avaliação do modelo de características foi de 14 minutos. A taxa de concordância entre as respostas dos participantes foi de 6,89% (2 de 29 perguntas). Houve grande variação nas respostas de um participante em relação aos demais (possível *outlier*). Desconsiderando as respostas desse participante, a taxa de concordância alcança 68,96% (20 do total de 29 perguntas).

A Figura 36 apresenta os percentuais das respostas em relação à avaliação do modelo de características.

Figura 36 - Percentual de respostas favoráveis, desfavoráveis e não aplicáveis.



Fonte: O autor, 2023.

Os percentuais das respostas dos participantes por categoria são apresentados na Tabela 10. Como se verifica, o participante 3 apresenta uma grande variação em relação aos demais participantes.

Tabela 10 - Respostas dos participantes por categoria.

Respostas		**Participantes**			
Descrição	**Total**	**P1**	**P2**	**P3**	**P4**
Favoráveis	82	34,15%	34,15%	4,88%	26,83%
Desfavoráveis	8	12,50%	12,50%	37,50%	37,50%
Não aplicáveis	26	0%	0%	84,62%	15,38%

Fonte: O autor, 2023.

Após a análise das observações dos participantes na avaliação do modelo de características, que consta, na íntegra, no Apêndice D, destacam-se os seguintes pontos-chave:

1. Observações gerais:

   - O participante 3 destacou que o modelo possui características relevantes, mas que ainda necessita de ajustes e melhorias; e
   - O participante 3 observou que algumas características de MS estão sobrepostas às características de ambientes de operação.

2. Clareza e correção das características do modelo:

   - O participante 3 observou que algumas características não estão relacionadas ao MS em si, mas à infraestrutura da AMS; e

- O participante 4 questionou a Distributividade ser uma característica, com a justificativa de ser inerente à AMS.

3. Opcionalidade/obrigatoriedade das características:

   - O participante 3 destacou que a maioria das características é desejável, ao invés de obrigatória; e
   - O participante 4 apontou não existir explicação sobre a definição da classificação da característica, como obrigatória, opcional ou alternativa.

4. Identificação do tipo das características:

   - O participante 4 considerou, no geral, ser possível identificar o tipo de cada característica a partir de sua descrição.

5. Representação de características relacionadas a técnicas de implementação:

   - O participante 4 apresentou dúvida na resposta deste item.

6. Características fora do escopo do domínio:

   - A inclusão da característica Distributividade no modelo foi questionada pelo participante 4, pois foi considerada inerente à AMS.

7. Escolha de características dentro de um conjunto (OU, OU Exclusivo):

   - Segundo o participante 4, há uma percepção de que as situações de escolha são adequadas, no entanto a existência do relacionamento obrigatório/opcional dentro de um relacionamento alternativo foi considerada confusa.

8. Relacionamentos não informados:

   - A ausência do relacionamento entre Modularidade e Manutenibilidade foi apontada pelo participante 1.

9. Hierarquia entre características:

   - Foi sugerido pelo participante 2 que a Disponibilidade deveria estar relacionada à Elasticidade e não diretamente à Escalabilidade;
   - O fato da Elasticidade estar no mesmo nível de Desempenho e Disponibilidade foi questionado pelo participante 3. Na opinião do participante, a Elasticidade é vista como um modo de obter/manter o Desempenho e a Disponibilidade, utilizando recursos necessários; e
   - O participante 4 apresentou dúvida quanto a troca de lugar no modelo envolvendo as características Escalabilidade e Desempenho.

10. Relacionamentos de agregação e composição:

    - O participante 4 informou não ser possível avaliar o item, pela não indicação dos relacionamentos de agregação e de composição no modelo.

11. Aplicabilidade das dependências ou relações de mútua exclusividade entre características no modelo:

    - O participante 4 observou que a semântica do relacionamento no modelo é do tipo “OR”, mas a legenda indica “XOR”.

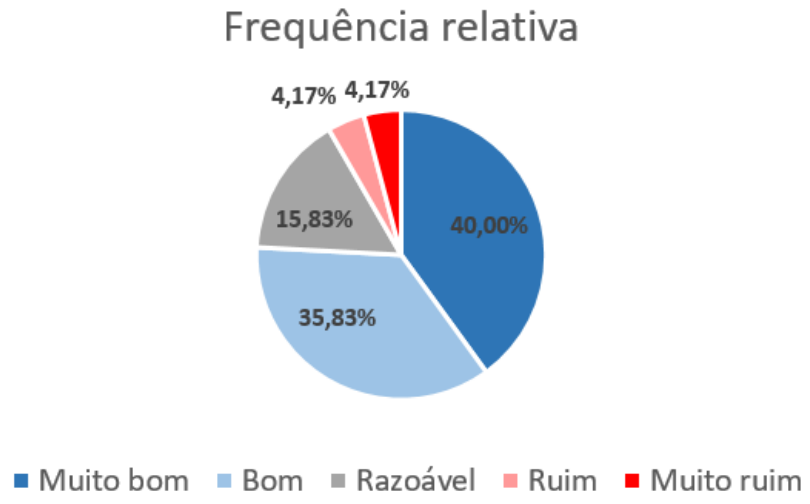
12. Adequação do modelo para orientar sua implementação:

    - O modelo foi considerado adequado pelo participante 4 para fornecer uma visão geral, mas não apresenta detalhes para orientar a implementação.

5.3.2 Glossário de termos

O tempo médio da avaliação do glossário de termos foi de 28 minutos. Cada item do glossário foi avaliado quanto ao grau de pertinência para explicar a característica associada. A Figura 37 apresenta as frequências relativas às respostas dos participantes na avaliação do glossário de termos.

Figura 37 - Frequência relativa das respostas dos participantes.

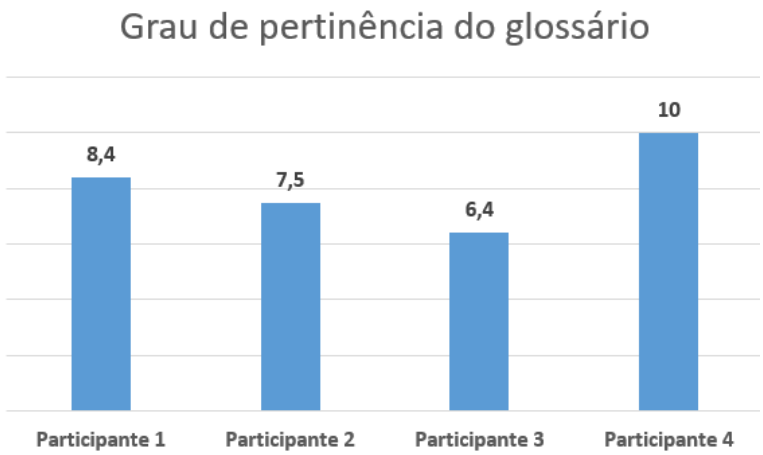


Fonte: O autor, 2023.

Verifica-se o alcance de aproximadamente 76% das avaliações dos participantes considerando as respostas “Muito bom” e “Bom”. A avaliação geral de todos os itens do glossário pelos participantes foi de 8,1, considerando a faixa de 0 a 10. Destaca-se que o

participante 4 avaliou o grau de pertinência de todos os itens do glossário como “Muito bom”, tratando-se de uma hipótese de ocorrência de fadiga na execução da avaliação. A Figura 38 apresenta a avaliação do grau de pertinência dos itens do glossário na perspectiva dos participantes.

Figura 38 - Grau de pertinência do glossário.



Fonte: O autor, 2023.

Após a análise das observações dos participantes na avaliação do glossário de termos, que consta, na íntegra, no Apêndice E, os seguintes pontos-chave são destacados:

1. Densidade das definições: Foi destacado pelo participante 2 que o texto de conceituação de algumas características é denso e cansativo de ser lido;

2. Relacionamentos hierárquicos e horizontais: Foi observado pelo participante 2 que os relacionamentos explorados são predominantemente hierárquicos, com características que contribuem para outras no sentido de baixo para cima. No entanto, há relacionamentos horizontais entre as características que não estão representados;

3. Quanto à Elasticidade: O participante 1 sugere que a característica vai além da capacidade de replicação dos MS;

4. Quanto ao Desempenho: No texto de definição da característica, foi sugerida, pelo participante 1, a substituição de REST (*Representational State Transfer*) por gRPC (*Google Remote Procedure Call*), como exemplo de mecanismo que possui maior ênfase na otimização da comunicação entre MS;

5. Quanto à Disponibilidade:

    - Foi observado pelo participante 1 que a capacidade de replicação por si só não garante a Disponibilidade. Outros aspectos, como a dependência de sistemas

externos e mecanismos de resiliência (*e.g.*, *circuit breaker*), desempenham um papel importante;

- Foi apontado pelo participante 2 que essa característica remete à Elasticidade; e
- O participante 3 discordou sobre o incremento do número de servidores ampliar os pontos de ataque e falhas. Para o participante, a replicação de um MS não altera esses componentes, uma vez que eles possuem exatamente a mesma base de código (*code base*).

6. Quanto à Descoberta de serviço: O participante 1 sugeriu o acréscimo na definição da característica de que a utilização de mecanismos de *service discovery* não é necessária, embora seja útil e considerada uma boa prática. Uma alternativa é a utilização de DNS interno à rede de MS, mecanismo que é fornecido pelo Kubernetes;

7. Quanto à Continuidade:

   - Foi destacado pelo participante 1 que a Continuidade vai, além das prátias de integração contínua (IC) e entrega contínua (EC), abrangendo diversas técnicas e métodos de continuidade no desenvolvimento de *software* (*Continuous Software Engineering*); e
   - O participante 2 observou a inexistência de relacionamento com Implantabilidade.

8. Quanto à Observabilidade: De acordo com o participante 3, o texto de definição deve ser mais detalhado, pois se trata de uma das principais características de MS;

9. Quanto à Testabilidade: Foi sugerida a inclusão de exemplos de tipos de testes relacionados a MS pelo participante 3;

10. Quanto à Resiliência: Foi sugerido pelo participante 1 acréscimo no texto de definição sobre a existência de padrões de arquitetura que aumentam a Resiliência de MS, como o *circuit breaker*, *timeout pattern* e o *retry pattern*;

11. Quanto à Segurança: Foi sugerido pelo participante 3 acréscimo no texto de definição, devido a essa característica apresentar diferentes aspectos em relação a MS;

12. Quanto à Orquestração: O participante 3 destaca que um dos principais motivos da adoção de MS é o serviço fornecido pelos orquestradores, motivo pelo qual esse item deve ser mais detalhado;

13. Quanto ao Tamanho:

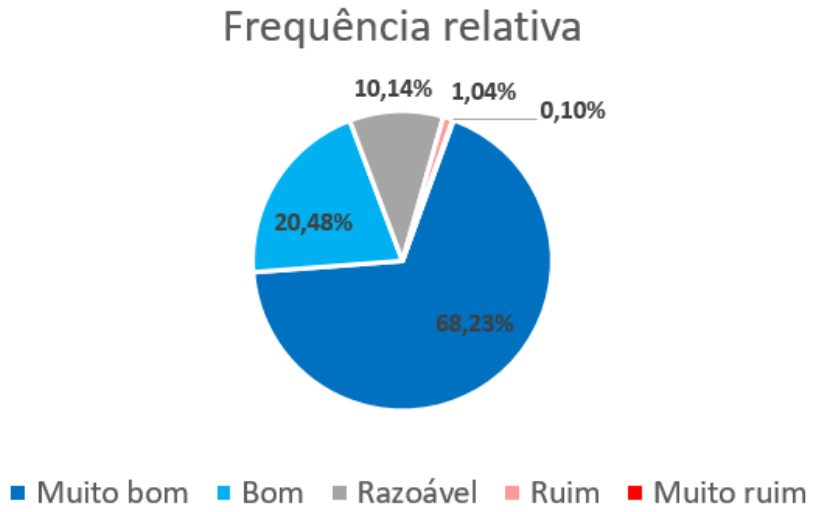
- Foi apontado pelo participante 1 que os benefícios em termos de manutenibilidade e extensibilidade, em relação ao pequeno tamanho de um MS, deve ser melhor explicado, dada a importância dessa característica; e
- O participante 3 apontou que a importância dessa característica não ficou evidente por meio da definição utilizada.

14. Quanto à Propriedade dos dados: O participante 1 relatou dúvida em relação a possibilidade de um MS utilizar ou não banco de dados; e
15. Quanto ao Balanceamento de carga: Foi destacado pelo participante 3 que essa característica faz parte do orquestrador.

5.3.3 Questionário de avaliação

O tempo médio da avaliação do questionário foi de 36 minutos. As perguntas do questionário foram avaliadas quanto ao grau de pertinência de cada pergunta para avaliar a característica associada. A Figura 39 apresenta as frequências relativas às respostas dos participantes na avaliação do questionário de avaliação.

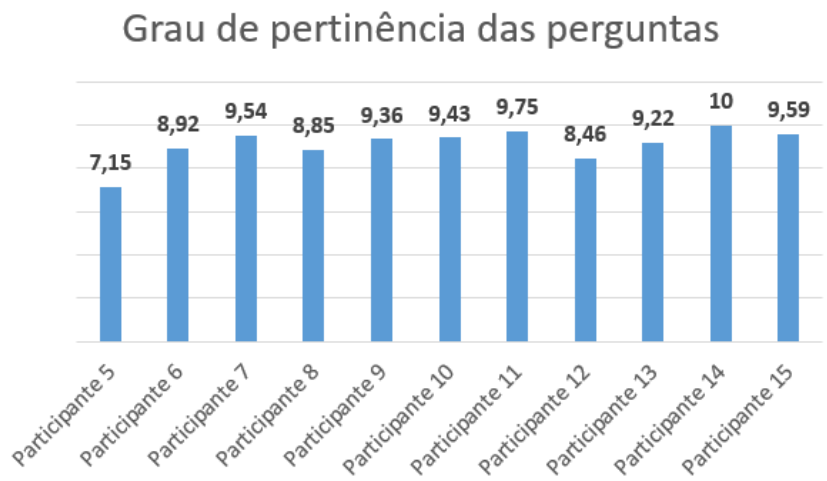
Figura 39 - Frequência relativa das respostas dos participantes.



Fonte: O autor, 2023.

Verifica-se o alcance de aproximadamente 88,71% das avaliações dos participantes considerando as respostas “Muito bom” e “Bom”. A avaliação geral de todas as perguntas do questionário pelos participantes foi de 9,11 considerando a faixa de 0 a 10. A Figura 40 apresenta a avaliação do grau de pertinência das perguntas do questionário na perspectiva dos participantes.

Figura 40 - Grau de pertinência do questionário.



Fonte: O autor, 2023.

O participante 14 avaliou todas as 87 perguntas do questionário como "muito bom". O número elevado de perguntas compondo o questionário de avaliação, reflexo da complexidade da AMS, pode ter ocasionado fadiga no participante. Para fins de conferência, foi observada a duração dessa avaliação, que foi de 29 minutos e 32 segundos. A média geral das respostas do questionário é 4,56.

Após a análise das observações dos participantes na avaliação do questionário, que consta, na íntegra, no Apêndice F, os seguintes pontos-chave são destacados:

1. Observações gerais:

    - Os participantes 7 e 10 consideraram o questionário muito extenso;
    - O participante 7 sugeriu a flexibilização do número de características a serem avaliadas, para o questionário ser utilizado como um DoR (*Definition of Ready*) para a arquitetura;
    - O participante 15 destacou a ausência de perguntas sobre o tipo de hospedagem do MS (*e.g.*, nuvem interna ou pública) referente à característica Virtualização; e
    - O participante 15 observou a existência de perguntas sobre IaaS (*Infrastructure as a Service*) e CaaS (*Container-as-a-Service*), mas a ausência de perguntas sobre SaaS (*Software as a Service*) e PaaS (*Platform-as-a-Service*).

2. Observações específicas:

    - Quanto à Escalabilidade: Referente à pergunta 1, que avalia essa característica pela perspectiva da demanda de recursos, foi sugerido pelo participante 9 considerar não apenas a adição de recursos (*e.g.*, processamento ou memória), mas também a remoção de recursos quando não forem mais necessários;

- Quanto à Manutenibilidade: Referente à pergunta 28, que avalia essa característica pela perspectiva da modificabilidade, o participante 15 manifestou incerteza quanto à pertinência da pergunta, destacando que qualquer aplicação deveria ser aberta a correções e a mudanças, mas destacou as aplicações adquiridas como pacotes fechados como contextos em que a alteração ou a evolução possa apresentar dificuldades associadas; e
- Quanto ao Acoplamento: Referente à pergunta 81, que avalia essa característica pela perspectiva do encapsulamento, o participante 11 ressaltou que a pergunta está mais relacionada ao conceito de baixo acoplamento.

5.3.4 Técnica de visualização

O tempo médio da avaliação foi de 1 minuto e 30 segundos. A avaliação do participante 16 teve a duração de trinta e oito segundos, enquanto a do participante 19, vinte e três segundos. A dos outros quatro participantes durou mais de um minuto e vinte e sete segundos. Supõe-se que os participantes 16 e 19 possam não ter interagido o suficiente com a técnica de visualização disponibilizada. Embora a duração dessas avaliações tenha sido curta, seus *feedbacks* foram importantes e, por isso, optou-se por manter suas avaliações. Nessa avaliação, os áudios dos participantes, obtidos por meio do protocolo *think-aloud*, foram transcritos para o formato texto. Esses dados foram separados em unidades de significado e categorizados, conforme o Apêndice G.

Os seguintes aspectos positivos foram destacados:

1. Avaliação da ferramenta: O participante 18 destacou a legibilidade, a relação com os requisitos conceituais de MS, a organização da estrutura e a facilidade de uso da ferramenta;

2. Relevância das métricas: O participante 16 considerou as métricas da técnica de visualização interessantes;

3. Satisfação com a visualização do resultado da avaliação: Os participantes 17, 18, 19 e 21 ficaram satisfeitos com a forma de visualização dos resultados das avaliações, por meio dos *gauges*;

4. Reação positiva à avaliação: O participante 21 reagiu positivamente à avaliação, achando-a interessante e destacando a utilidade dos *gauges*;

5. Rapidez e clareza na mensuração dos atributos: O participante 17 destacou que a visualização permitiu uma mensuração rápida e concisa das características dos MS;

6. Identificação de estados satisfatórios e mínimos: O participante 17 destacou que a visualização possibilitou identificar se uma característica atingiu um estado satisfatório ou ideal;

7. Identificação de oportunidades de melhorias: Segundo os participantes 18, 19 e 21, os *gauges* ajudam a identificar características que podem ser melhoradas;

8. Potencial para expansão: O participante 19 destacou que a técnica de visualização apresenta potencial para fornecer mais informações;

9. Recomendação para adoção da visualização: O participante 17 recomendou a adoção da técnica de visualização para acompanhamento dos processos de desenvolvimento e manutenção de MS; e

10. Valor da avaliação para profissionais de TI: Segundo o participante 21, a avaliação foi considerada importante para profissionais que trabalham no nível arquitetural.

Os seguintes aspectos negativos foram indicados:

1. Detalhamento excessivo: O participante 16 observou que a técnica de visualização apresenta detalhes demais, os quais na prática não são utilizados;

2. Incerteza sobre a aplicabilidade geral: A aplicabilidade em um contexto geral foi questionada pelo participante 16, pois há MS que acessam serviços de terceiros, os quais não se tem acesso ao código-fonte; e

3. Discrepância na avaliação entre características genéricas e específicas: Os participantes 18 e 20 relataram que itens e subitens não aparentam ter uma relação de peso ou influência clara entre si, afetando a interpretação do resultado dos *gauges*.

As seguintes sugestões de melhorias foram apresentadas:

1. Avaliação de características genéricas e específicas: O participante 20 sugeriu que características genéricas não deveriam apresentar perguntas, sendo sua pontuação calculada com a média da pontuação das características específicas; e

2. Inclusão de opção de resposta na avaliação: O participante 21 reportou que algumas perguntas não eram aplicáveis ao cenário sendo avaliado e que, para esses casos, deveria estar disponível a opção “não aplicável” como possibilidade de resposta.

5.4 **Ameaças à validade**

De acordo com Wohlin et al. (2000), ameaças à validade são fatores que podem comprometer a validade de um estudo ou pesquisa. Essas ameaças podem surgir em diferentes etapas do processo de pesquisa, desde o planejamento até a interpretação dos resultados, e podem afetar a confiabilidade e a generalização dos resultados obtidos.

Não há como garantir que essas ameaças não afetem os resultados, mesmo com a tentativa de evitá-las. Assim, conforme proposto por Wohlin et al. (2000), as ameaças identificadas foram categorizadas em validade interna, validade externa, validade de conclusão e validade de construção:

1. Ameaças à validade interna (podem prejudicar o conhecimento sobre o relacionamento de causa e efeito entre o tratamento e o resultado):

    - Como as avaliações foram realizadas remotamente, de acordo com a disponibilidade do participante, não é possível confirmar se as circunstâncias eram as mesmas nas ocasiões em que cada participante colaborou com o estudo;
    - Como não se trata de um estudo observacional, considerou-se que os participantes seguiram as instruções e a ordem das atividades; e
    - O desempenho de alguns participantes pode ter sido afetado pela avaliação de alguns artefatos, como o questionário de avaliação, que foi considerado muito extenso.

2. Ameaças à validade externa (limitam a generalização dos resultados do estudo para outros contextos fora do ambiente avaliado):

    - A reprodutibilidade pode ser considerada limitada, devido à amostra, por conveniência, dos trabalhos utilizados na atividade analisar trabalhos da metodologia apresentada no Capítulo 3; e
    - A representatividade dos participantes é limitada, o que se deve, em parte, à dificuldade em encontrar especialistas disponíveis para compor a amostra. Pelo fato de a amostra utilizada ser não probabilística, não foi possível determinar a *priori* o tamanho da população e o número total esperado de participantes (JERONIMO Jr., 2023).

3. Ameaças à validade de conclusão (fatos que prejudicam o estabelecimento de relacionamentos estatísticos entre o tratamento e o resultado):

    - Não foram utilizados testes estatísticos para análise dos dados, pois os tamanhos das amostras são bastante limitados. Com isso, os resultados do estudo não são conclusivos, fornecendo apenas alguns indícios (OLIVEIRA, 2011).

4. Ameaças à validade de construção (eventos que podem impedir que a configuração do experimento reflita adequadamente a construção do relacionamento entre o tratamento e o resultado):

    - Não foi realizada comparação com outras ferramentas existentes. Para estabelecer a validade de construção de uma nova ferramenta de visualização, é importante validar sua eficácia em comparação com outras ferramentas existentes. A ausência de uma comparação adequada pode dificultar a interpretação dos resultados e limitar a compreensão do valor agregado pela nova ferramenta (HEER; SHNEIDERMAN, 2012).

5.5 **Conclusão**

Neste capítulo, procedeu-se à avaliação dos artefatos, incluindo o modelo de características, o glossário de termos, o questionário de avaliação e a técnica de visualização. As avaliações foram conduzidas para determinar a aplicabilidade desses artefatos, assim como o nível de concordância entre pesquisadores e profissionais no contexto da engenharia de *software*.

O principal benefício esperado com a utilização dessa abordagem é auxiliar arquitetos e desenvolvedores de *software* na compreensão dos diferentes interesses (*concerns*) não-funcionais associados à AMS, de forma a melhorar o entendimento e a avaliação de seus MS. Esse benefício foi percebido de acordo com os resultados do experimento, que forneceram indícios de que a utilização da abordagem InA^{2}rMS pode contribuir em atividades relacionadas aos processos de construção e de manutenção de MS. Todavia, essa afirmação só pode ser confirmada após a realização de novos estudos.

No próximo capítulo, serão apresentadas as considerações finais e conclusões deste trabalho, além de suas contribuições, limitações e trabalhos futuros.

CONSIDERAÇÕES FINAIS

A AMS ficou em evidência na última década, devido à sua adequabilidade às tecnologias nativas de nuvem e à sua natureza distribuída, podendo prover diversos benefícios para as organizações. No entanto, devido à sua complexidade, seu desenvolvimento requer uma compreensão clara das características desejadas e dos recursos necessários para implementá-las, tornando-se um desafio para arquitetos de *software*.

Diante desse cenário, a modelagem das principais características no contexto de MS, com a captura de suas semelhanças e de suas variabilidades, com a definição de suas relações estruturais e de suas interdependências, pode subsidiar um instrumento de avaliação do atendimento a essas características.

Nesse contexto, o presente trabalho propõe a abordagem InA^{2}rMS, uma ferramenta para a análise e avaliação de MS, que é composta por um modelo de características, um glossário de termos e um questionário de avaliação integrado a uma técnica de visualização de dados, permitindo a avaliação do atendimento às características presentes no modelo de características.

Considerando a abordagem proposta em relação aos trabalhos relacionados, apresentados no Capítulo 2, os seguintes pontos são destacados:

- Diante do *survey* realizado por Mattsson, Grahn e Mårtensson (2006) e da revisão sistemática realizada por Barcelos (2006), sobre métodos de avaliação arquitetural, um dos grandes diferenciais dessa proposta é a possibilidade de realizar uma avaliação considerando um conjunto abrangente de características de forma simultânea sem o custo elevado, como ressaltado por Barcelos (2006);

- As abordagens propostas por Lehmann e Sandnes (2017), Engel et al. (2018), Cardarelli et al. (2019), Cojocaru, Oprescu e Uta (2019), Rosa et al. (2020), Fourati, Marzouk e Jmaiel (2021) apresentam, individualmente, um conjunto pouco abrangente em relação às características existentes na AMS;

- Em relação à taxonomia proposta por Garriga (2018), a abordagem proposta se destaca por fornecer, além de um modelo de características, um instrumento de análise e avaliação de MS, que é composto por um glossário de termos e um questionário de avaliação integrado a uma técnica de visualização de dados. Além disso, a taxonomia referencia tecnologias e ferramentas específicas, o que, com o tempo, pode tornar-se defasada, o que também foi identificado na classificação proposta por (FOURATI; MARZOUK; JMAIEL, 2021);

- Em relação às propostas dos trabalhos de Söylemez, Tekinerdogan e Tarhan (2022a) e de Söylemez, Tekinerdogan e Tarhan (2022b), ambas podem ser utilizadas como

fontes de consulta; a primeira, fornecendo soluções para as nove categorias de desafios encontrados na adoção da AMS; e a segunda, com o mapeamento das características com as tecnologias e os serviços da AMS; e

- Destaca-se que vinte e quatro características presentes no modelo de características proposto são utilizadas nos estudos apresentados no Capítulo 2, fornecendo indícios de que foram selecionadas características relevantes da AMS.

Contribuições

As contribuições deste trabalho estão relacionadas às questões de pesquisa apresentadas a seguir:

QP.1 Quais características são relevantes da AMS?

A questão de pesquisa é respondida com a criação do modelo de características apresentado no Apêndice A. As características identificadas foram relacionadas aos respectivos trabalhos de modo a permitir o entendimento do nível de importância de cada característica.

QP2. Como as características relevantes da AMS se relacionam?

Essa questão de pesquisa é respondida com o modelo de características proposto, apresentado no Apêndice A. Cada característica do modelo, assim como seus relacionamentos de interdependência, são apresentados no Apêndice B.

QP3. Como analisar e avaliar microsserviços em relação às características da AMS?

Essa questão é respondida com a elaboração do questionário de avaliação referente às características presentes no modelo de características. O questionário é apresentado no Apêndice C.

QP4. Como expressar o grau de atendimento de microsserviços às características de uma forma intuitiva e precisa?

Essa questão de pesquisa é respondida por meio da criação do questionário de avaliação integrado à técnica de visualização, fornecendo uma avaliação simples e precisa, para expressar o grau de atendimento às características presentes no modelo de características.

Além dessas contribuições, a metodologia apresentada no Capítulo 3 pode ser reutilizada em estudos envolvendo novas arquiteturas.

Durante o desenvolvimento desse trabalho, foi possível observar que a abordagem proposta pode oferecer contribuições significativas para arquitetos e desenvolvedores de *software*, especialmente nas atividades relacionadas à avaliação, construção e manutenção de MS. Esses profissionais podem se beneficiar ao compreender as diversas dimensões e preocupações associadas ao desenvolvimento de *software* utilizando tal arquitetura.

Limitações

Algumas limitações foram identificadas durante o desenvolvimento deste trabalho. Destas limitações, destacam-se:

- O modelo de características proposto teve como base as características selecionadas de acordo com a revisão da literatura apresentada no Capítulo 3. Este conjunto de características selecionado possui relação com a amostra, por conveniência, de trabalhos utilizados. Caso essa amostra seja modificada, o conjunto de características pode ser alterado;
- No modelo de caraterísticas avaliado pelos participantes, que consta no Apêndice A, a legenda utilizada para representar o agrupamento de características do tipo "OR" encontra-se inconsistente com relação a semântica esperada, o que foi apontado por um participante. A semântica deve ser alterada para refletir esse apontamento;
- Em virtude do tamanho da amostra obtida, não foi possível realizar a avaliação da confiabilidade e consistência interna do questionário de avaliação por meio de técnicas como o coeficiente *alfa* de Cronbach (HORA; TORRES; ARICA, 2010); e
- Conforme a Subseção 3.2.1, o número elevado de perguntas compondo o questionário de avaliação, reflexo da complexidade da AMS, pode ter ocasionado fadiga nos participantes.

Trabalhos futuros

A revisão da literatura realizada proporcionou a identificação de um número abrangente de características consideradas relevantes sobre a AMS.

A elaboração e o resultado final do modelo de características ratificam a complexidade do tópico, ressaltando a necessidade de uma estruturação que possa ser consultada em trabalhos existentes e futuros.

Essa base conceitual fornecida é o primeiro passo para a realização de estudos com foco em características específicas.

A condução de revisões sistemáticas relacionadas às características poderia ampliar as informações de cada característica, complementando o presente estudo. Assim, a dissertação de Stutzel (2020), por exemplo, que apresenta foco na característica escalabilidade, poderia fornecer métricas relevantes para a elaboração de perguntas para a composição do questionário de avaliação.

Além disso, a contribuição desse trabalho pode ser estendida para suportar a avaliação utilizando outros parâmetros (*e.g.*, métricas). Assim, o questionário de avaliação

poderia ser aprimorado para apoiar o planejamento de capacidade (*capacity planning*).

O processo de avaliação dos artefatos pelos participantes proporcionou os seguintes *feedbacks* importantes para o processo de melhoria da abordagem InA^{2}rMS:

- Os relacionamentos de interdependência entre as características presentes no modelo de características, conforme Apêndice B, fornecem informações preliminares para o estudo futuro das *constraints* do modelo de características proposto;
- A inclusão da opção de resposta "não se aplica" no questionário de avaliação tornaria esse artefato mais abrangente a diversos cenários;
- A possibilidade de seleção de características a serem avaliadas tornaria o questionário flexível a diferentes contextos;
- A técnica de visualização poderia ser alterada para disponibilizar opções em sua *interface* para aumentar e diminuir detalhes, como os relacionamentos horizontais entre as características; e
- Na visualização do resultado da avaliação, a pontuação das características genéricas poderia levar em consideração a pontuação das características específicas.

REFERÊNCIAS

ABDELFATTAH, Amr S.; CERNY, Tomas. Roadmap to reasoning in microservice systems: A rapid review. *Applied Sciences*, v. 13, n. 3, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/3/1838>.

ABRAMS, Charles; SCHULTE, Roy W. Service-oriented architecture overview and guide to soa research. *Gartner Research*, 2008.

AL-DEBAGY, Omar; MARTINEK, Peter. A comparative review of microservices and monolithic architectures. In: *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*. [S.l.: s.n.], 2018. p. 000149–000154.

AL-DEBAGY, Omar; MARTINEK, Péter. Extracting microservices' candidates from monolithic applications: Interface analysis and evaluation metrics approach. In: *2020 IEEE 15th International Conference of System of Systems Engineering (SoSE)*. [S.l.: s.n.], 2020. p. 289–294.

ALTURKI, Fadil; KHÉDRI, Ridha. A tool for formal feature modeling based on BDDs and product families algebra. In: . [S.l.: s.n.], 2010.

ARAUJO, Elena Augusta. *Uma Abordagem de Conformidade Arquitetural para Arquitetura de Microsserviços*. Dissertação (Mestrado) — Universidade Federal de Lavras - UFLA, 2019.

ASSELDONK, Lars van. *From a Monolith to Microservices: the Effect of Multi-view Clustering*. Dissertação (Mestrado) — Utrecht University, 2021. Disponível em: <https://studenttheses.uu.nl/handle/20.500.12932/148>.

AWS. *Microservices architecture on AWS*. 2022. Online; accessed 19 December 2022. Disponível em: <https://docs.aws.amazon.com/pt_br/whitepapers/latest/microservices-on-aws/simple-microservices-architecture-on-aws.html>.

BABAR, M.A.; ZHU, L.; JEFFERY, R. A framework for classifying and comparing software architecture evaluation methods. In: *2004 Australian Software Engineering Conference. Proceedings*. [S.l.: s.n.], 2004. p. 309–318.

BALALAIE, Armin et al. Microservices migration patterns. *Software: Practice and Experience*, v. 48, n. 11, p. 2019–2042, 2018. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2608>.

BAO, Fan; CHEN, Jia. Visual framework for big data in d3.js. In: *2014 IEEE Workshop on Electronics, Computer and Applications*. [S.l.: s.n.], 2014. p. 47–50.

BARBACCI, Mario et al. *Quality Attributes*. Pittsburgh, PA, 1995. Disponível em: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=12433>.

BARBOSA, Marx Haron Gomes. *A process to migrate legacy systems with business rules contained in stored procedures to a microservice-oriented architecture*. Dissertação (Mestrado) — Universidade Estadual do Ceará, 2020.

BARCELOS, Rafael Ferreira. *Uma abordagem para inspeção de documentos arquiteturais baseada em checklist*. Dissertação (Mestrado) — UFRJ/COPPE, 2006.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. *Software Architecture in Practice (2nd Edition)*. [S.l.]: Addison-Wesley Professional, 2003.

BECK, Kent. *Extreme Programming Explained: Embrace Change*. USA: Addison-Wesley Longman Publishing Co., Inc., 1999. ISBN 0201616416.

BECKER, Alex Malmann. *O impacto do uso de micro serviços na evolução de uma linha de produto de software*. Dissertação (Mestrado) — Universidade Federal de São Carlos - UFSCar, 2019.

BERG, Tom van den; SIEGEL, Barry; CRAMP, Anthony. Containerization of high level architecture-based simulations: A case study. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, v. 14, 09 2016.

BITTES, José Márcio; SANTORO, Flávia Maria; BORGES, Marcos. A implantabilidade como atributo de qualidade do software. In: *Anais do IV Simpósio Brasileiro de Qualidade de Software*. Porto Alegre, RS, Brasil: SBC, 2005. p. 218–231. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbqs/article/view/16165>.

BOGADO, Verónica; GONNET, Silvio; LEONE, Horacio. Devs-based methodological framework for multi-quality attribute evaluation using software architectures. In: *2017 XLIII Latin American Computer Conference (CLEI)*. [S.l.: s.n.], 2017. p. 1–10.

BOSTOCK, Michael; OGIEVETSKY, Vadim; HEER, Jeffrey. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, v. 17, n. 12, p. 2301–2309, 2011.

BOUCKÉ, Nelis; HOLVOET, Tom. Dealing with concerns ask for an architecture-centric approach. 01 2005.

BRILHANTE, Jonathan Lincoln Gandhi Andrade Pires. *Uma abordagem para construção de microsserviços reativos baseadas em filas assíncronas*. Dissertação (Mestrado) — Universidade Federal da Paraíba (UFPB/PPGI), https://repositorio.ufpb.br/jspui/handle/123456789/13820, 2018.

CARDARELLI, Mario et al. An extensible data-driven approach for evaluating the quality of microservice architectures. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2019. (SAC '19), p. 1225–1234. ISBN 9781450359337. Disponível em: <https://doi-org.ez83.periodicos.capes.gov.br/10.1145/3297280.3297400>.

CARVALHO, Luiz et al. Analysis of the criteria adopted in industry to extract microservices. In: *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*. [S.l.: s.n.], 2019. p. 22–29.

CHEN, C.; HÄRDLE, W.K.; UNWIN, A. *Handbook of Data Visualization*. Springer Berlin Heidelberg, 2007. (Springer Handbooks of Computational Statistics). ISBN 9783540330370. Disponível em: <https://books.google.com.br/books?id=zzCiSJoohuQC>.

CHI, Li-Jung; HUANG, Chi-Hsuan; CHUANG, Kun-Ta. Mobile-friendly and streaming web-based data visualization. In: *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. [S.l.: s.n.], 2016. p. 124–129.

CLEMENTS, Paul et al. *Documenting Software Architectures: Views and Beyond*. Second. [S.l.]: Addison-Wesley Professional, 2010.

CLEMENTS, P. et al. Documenting software architectures: views and beyond. In: *25th International Conference on Software Engineering, 2003. Proceedings.* [S.l.: s.n.], 2003. p. 740–741.

CLOUD, Google. *Microservices architecture on Google Cloud*. 2021. Online; accessed 19 December 2022. Disponível em: <https://cloud.google.com/blog/topics/developers-practitioners/microservices-architecture-google-cloud>.

COJOCARU, Michel; OPRESCU, Ana-Maria; UTA, Alexandru. Attributes assessing the quality of microservices automatically decomposed from monolithic applications. In: . [S.l.: s.n.], 2019. p. 84–93.

COSTA, Marcelo de França. *DIRECTOR: A Cloud Microservice Selection Framework*. Tese (Doutorado) — UFRJ/COPPE, 2019.

de Sousa Neto, M. V. *Computação em Nuvem: Nova Arquitetura de TI*. [S.l.]: Brasport, 2015. ISBN 9788574527475.

DIRKSEN, Jos. *SOA Governance in Action*. 1st. ed. [S.l.]: Manning, 2012. ISBN 9781617290275.

ENGEL, Thomas et al. Evaluation of microservice architectures: A metric and tool-based approach. In: _____. [S.l.: s.n.], 2018. p. 74–89. ISBN 978-3-319-92900-2.

ERICSSON, K. Anders; SIMON, Herbert A. *Protocol Analysis: Verbal Reports as Data*. The MIT Press, 1993. ISBN 9780262272391. Disponível em: <https://doi.org/10.7551/mitpress/5657.001.0001>.

ERL, T. *SOA: Principios De Design De Serviços*. Pearson Prentice Hall, 2009. ISBN 9788576051893. Disponível em: <https://books.google.com.br/books?id=UurZPgAACAAJ>.

FAHMI, Faisal; HUANG, Pei Shu; WANG, Feng-Jian. A method to detecting artifact anomalies in a microservice architecture. In: . [S.l.: s.n.], 2020. p. 81–88. Publisher Copyright: © 2020 IEEE. Copyright: Copyright 2021 Elsevier B.V., All rights reserved.; 26th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2020 ; Conference date: 02-12-2020 Through 04-12-2020.

FEW, S. *Now You See it: Simple Visualization Techniques for Quantitative Analysis*. Analytics Press, 2009. ISBN 9780970601988. Disponível em: <https://books.google.com.br/books?id=xw_qOwAACAAJ>.

FOURATI, Mohamed Hedi; MARZOUK, Soumaya; JMAIEL, Mohamed. A review of container level autoscaling for microservices-based applications. In: *2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. [S.l.: s.n.], 2021. p. 17–22.

FOWLER, M. *Refactoring: Improving the Design of Existing Code*. [S.l.]: Addison-Wesley, 2019. (A Martin Fowler signature book). ISBN 9780134757599.

FOWLER, S.J. *Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software*. [S.l.]: Novatec Editora, 2017. ISBN 9788575226216.

FRACTALYTICS. *Visualization of scikit-learn Decision Trees with d3.js*. 2019. Online; accessed 11 Nov 2022. Disponível em: <http://fractalytics.io/visualization-scikit-learn-decision-trees-d3-js>.

FRANCESCO, Paolo Di. Architecting microservices. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. [S.l.: s.n.], 2017. p. 224–229.

FULLER, M. *Software Studies: A Lexicon*. Books24x7.com, 2008. (EBSCO ebook academic collection). ISBN 9780262062749. Disponível em: <https://books.google.com.br/books?id=LFJ3ashVBuIC>.

GAMMA, Erich et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1. ed. Addison-Wesley Professional, 1994. ISBN 0201633612. Disponível em: <http://www.amazon.com/Design-Patterns-Elements-Reusable-Object-Oriented/dp/0201633612/ref=ntt_at_ep_dpi_1>.

GARRIGA, Martin. Towards a taxonomy of microservices architectures. In: _____. [S.l.: s.n.], 2018. p. 203–218. ISBN 978-3-319-74780-4.

GARRIGA, Martin et al. Web services composition mechanisms: A review. *IETE Technical Review*, v. 32, p. 1–8, 03 2015.

_____. Restful service composition at a glance: A survey. *Journal of Network and Computer Applications*, v. 60, p. 32–53, 2016. ISSN 1084-8045. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1084804515002933>.

GOKYER, Gokhan et al. Non-functional requirements to architectural concerns: Ml and nlp at crossroads. In: *2008 The Third International Conference on Software Engineering Advances*. [S.l.: s.n.], 2008. p. 400–406.

GORTON, Ian. *Essential Software Architecture (2. ed.)*. [S.l.: s.n.], 2011. ISBN 978-3-642-19175-6.

GOS, Konrad; ZABIEROWSKI, Wojciech. The comparison of microservice and monolithic architecture. In: *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. [S.l.: s.n.], 2020. p. 150–153.

GUEST, Greg; BUNCE, Arwen; JOHNSON, Laura. How many interviews are enough?: An experiment with data saturation and variability. *Field Methods*, v. 18, n. 1, p. 59–82, 2006. Disponível em: <https://doi.org/10.1177/1525822X05279903>.

HAAK, M.J. van den; JONG, M.D.T. de. Exploring two methods of usability testing: concurrent versus retrospective think-aloud protocols. In: *IEEE International Professional Communication Conference, 2003. IPCC 2003. Proceedings.* [S.l.: s.n.], 2003. p. 3 pp.–.

HANSEN, Poul Kyvsgaard; SUN, Hongyi. Complexity in managing modularization. In: *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*. [S.l.: s.n.], 2011. v. 3, p. 537–540.

HEER, Jeffrey; SHNEIDERMAN, Ben. Interactive dynamics for visual analysis. *Communications of the ACM*, v. 55, p. 45–54, 04 2012.

HENDERSON, R. D. et al. A comparison of the four prominent user-based methods for evaluating the usability of computer software. *Ergonomics*, Taylor & Francis, v. 38, n. 10, p. 2030–2044, 1995. Disponível em: <https://doi.org/10.1080/00140139508925248>.

HORA, Henrique da; TORRES, Gina; ARICA, José. Confiabilidade em questionários para qualidade: Um estudo com o coeficiente alfa de cronbach. *Produto & Produção*, v. 11, 06 2010.

HUMBLE, Jez; FARLEY, David G. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. [S.l.]: Addison-Wesley, 2010. ISBN 978-0-321-60191-9.

JAMSHIDI, Pooyan et al. Microservices: The journey so far and challenges ahead. *IEEE Software*, v. 35, n. 3, p. 24–35, 2018.

JERONIMO Jr., Helvio. *isTDM: An Evidence-Based Framework to Support the Technical Debt Management in Software Projects*. Tese (Doutorado) — UFRJ/COPPE, 2023.

KANG, Kyo et al. Feature-oriented domain analysis (foda) feasibility study. 01 1990.

KAZANAVIČIUS, Justas; MAŽEIKA, Dalius. Migrating legacy software to microservices architecture. In: *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. [S.l.: s.n.], 2019. p. 1–5.

KIM, Gene et al. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. [S.l.]: IT Revolution Press, 2016. ISBN 1942788002.

KISTASAMY, Christopher; MERWE, Alta van der; HARPE, Andre De La. The relationship between service oriented architecture and enterprise architecture. In: *2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*. [S.l.: s.n.], 2010. p. 129–137.

KLOCK, Sander et al. Workload-based clustering of coherent feature sets in microservice architectures. In: *2017 IEEE International Conference on Software Architecture (ICSA)*. [S.l.: s.n.], 2017. p. 11–20.

KRUEGER, Charles W. Software reuse. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 24, n. 2, p. 131–183, jun 1992. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/130844.130856>.

KUMAR, Pradeep; SINGH, Shailendra Narayan; DAWRA, Sudhir. Software component reusability prediction using extra tree classifier and enhanced harris hawks optimization algorithm. *International Journal of System Assurance Engineering and Management*, v. 13, n. 2, p. 892–903, Apr 2022. ISSN 0976-4348. Disponível em: <https://doi.org/10.1007/s13198-021-01359-6>.

LAIGNER, Rodrigo et al. Data management in microservices: State of the practice, challenges, and research directions. *Proc. VLDB Endow.*, VLDB Endowment, v. 14, n. 13, p. 3348–3361, sep 2021. ISSN 2150-8097. Disponível em: <https://doi.org/10.14778/3484224.3484232>.

LAURETIS, Lorenzo De. From monolithic architecture to microservices architecture. In: *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. [S.l.: s.n.], 2019. p. 93–96.

LEE, Kwanwoo; KANG, Kyo; LEE, Jaejoon. Concepts and guidelines of feature modeling for product line software engineering. In: . [S.l.: s.n.], 2002. p. 62–77. ISBN 978-3-540-43483-2.

LEE, Sungchul; JO, Ju-Yeon; KIM, Yoohwan. Performance testing of web-based data visualization. In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2014. p. 1648–1653.

LEHMANN, Martin; SANDNES, Frode Eika. A framework for evaluating continuous microservice delivery strategies. In: *Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing*. New York, NY, USA: Association for Computing Machinery, 2017. (ICC '17). ISBN 9781450347747. Disponível em: <https://doi-org.ez83.periodicos.capes.gov.br/10.1145/3018896.3018961>.

LEHTOLA, Henrikki. *Effective Migration of an Automation System to Microservice Architecture*. Dissertação (Mestrado) — Tampere University, 2020. Disponível em: <https://trepo.tuni.fi/handle/10024/123002>.

LENARDUZZI, Valentina et al. Does migrating a monolithic system to microservices decrease the technical debt? *Journal of Systems and Software*, v. 169, 07 2020.

LEON, Freddy Tapia et al. From monolithic systems to microservices: A comparative study of performance. *Applied Sciences*, v. 10, p. 5797, 08 2020.

LEVITA, Carlos de Amorim. *Proposta de modelo para avaliação da maturidade DevOps: estudo de caso em empresas de grande porte*. Dissertação (Mestrado) — PUC-SP, 2017.

LEWIS, J.; FOWLER, M. *Microservices: A definition of this new architectural term*. 2014. Online; accessed 31 May 2022. Disponível em: <https://martinfowler.com/articles/microservices.html>.

LIU, Guozhi et al. Microservices: architecture, container, and challenges. In: *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. [S.l.: s.n.], 2020. p. 629–635.

LUO, Diaohan et al. On the knowledge graphs of postgraduate entrance english examination based on wordnet and d3.js. In: *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*. [S.l.: s.n.], 2020. v. 1, p. 991–996.

MAIA, Amanda Fortes Dalla Valle Majó da. *Representação gráfica de mapas para daltônicos : um estudo de caso dos mapas da rede integrada de transporte de Curitiba*. Dissertação (Mestrado) — Universidade Federal do Paraná. Setor de Artes, Comunicação

e Design. Programa de Pós-Graduação em Design, https://hdl.handle.net/1884/29947, 2013.

MARZULLO, Fabio Perez. *Serviços de Desenvolvimento de Software: Uma Abordagem para Reutilização de Modelos Conceituais*. Tese (Doutorado) — UFRJ/COPPE, 2014.

MATTSSON, Michael; GRAHN, Håkan; MÅRTENSSON, Frans. Software architecture evaluation methods for performance, maintainability, testability, and portability. In: . [S.l.: s.n.], 2006.

MEEKS, E. *D3.js in Action*. Manning, 2015. ISBN 9781617292118. Disponível em: <https://books.google.com.br/books?id=40mOoAEACAAJ>.

MELLO, Rafael de et al. Verification of software product line artefacts: A checklist to support feature model inspections. *JOURNAL OF UNIVERSAL COMPUTER SCIENCE*, v. 20, p. 720–745, 01 2014.

MERINO, Leonel et al. A systematic literature review of software visualization evaluation. *Journal of Systems and Software*, v. 144, 06 2018.

MICROSOFT. *Projetar comunicação entre serviços para microsserviços*. 2022. Disponível em: <https://docs.microsoft.com/pt-br/azure/architecture/microservices/design/interservice-communication>. Acesso em: 25 de agosto de 2022.

MUNAF, Raja Mubashir et al. Microservices architecture: Challenges and proposed conceptual design. In: *2019 International Conference on Communication Technologies (ComTech)*. [S.l.: s.n.], 2019. p. 82–87.

NEWMAN, S. *Building Microservices*. [S.l.]: O'Reilly Media, 2015. ISBN 9781491950357.

_____. *Criando Microsserviços – 2a Edição: Projetando sistemas com componentes menores e mais especializados*. [S.l.]: Novatec Editora, 2022. ISBN 9786586057898.

O'CONNOR, Rory V.; ELGER, Peter; CLARKE, Paul M. Continuous software engineering—a microservices architecture perspective. *Journal of Software: Evolution and Process*, v. 29, n. 11, p. e1866, 2017. E1866 JSME-16-0193.R2. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1866>.

OLIVEIRA, Marcelo. *PREViA: Uma Abordagem para a Visualização da Evolução de Modelos de Software*. Dissertação (Mestrado) — UFRJ/COPPE, 02 2011.

OLIVEIRA, Marcelo Schots de. *On The Use of Visualization for Supporting Software Reuse*. Tese (Doutorado) — UFRJ/COPPE, 2015.

PONTES, Danielle Pompeu Noronha. *Evolução de software baseada em avaliação de arquiteturas*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais, 2012.

PRESSMAN, Roger S. *Software Engineering: A Practitioner's Approach*. 7th. ed. New York, NY, USA: McGraw-Hill, Inc., 2009. ISBN 0073375977.

RICHARDS, Mark. *Microservices vs. service-oriented architecture*. 2016. Online; accessed 31 May 2022. Disponível em: <https://www.oreilly.com/radar/microservices-vs-service-oriented-architecture/>.

ROMNEY, A. Kimball; WELLER, Susan C.; BATCHELDER, William H. Culture as consensus: A theory of culture and informant accuracy. *American Anthropologist*, v. 88, n. 2, p. 313–338, 1986. Disponível em: <https://anthrosource.onlinelibrary.wiley.com/doi/abs/10.1525/aa.1986.88.2.02a00020>.

ROSA, Thatiane de Oliveira et al. 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*. New York, NY, USA: Association for Computing Machinery, 2020. (EuroPLoP '20). ISBN 9781450377690. Disponível em: <https://doi.org/10.1145/3424771.3424809>.

ROST, Lisa Charlotte. *What to Consider When Choosing Colors for Data Visualization*. 2018. Online; accessed 13 March 2023. Disponível em: <https://www.dataquest.io/blog/what-to-consider-when-choosing-colors-for-data-visualization/>.

ROY, Banani; GRAHAM, T. C. Nicholas. Methods for evaluating software architecture: A survey. In: . [S.l.: s.n.], 2008.

SAMARPIT, Tuli. *Microservices vs SOA: What's the Difference?* 2018. Disponível em: <https://dzone.com/articles/microservices-vs-soa-whats-the-difference>. Acesso em: 21 de outubro de 2022.

SANTOS, Eduardo Fernandes Mioto de Oliveira dos. *Sugestão de Criticidade Baseada em Multicritério para Arquiteturas Estabelecidas Orientadas a Microsserviços*. Dissertação (Mestrado) — UFRJ/COPPE, 2020.

SANTOS, Eduardo Fernandes Mioto de Oliveira dos; WERNER, Claudia Maria Lima. A survey on microservices criticality attributes on established architectures. In: *2019 International Conference on Information Systems and Software Technologies (ICI2ST)*. [S.l.: s.n.], 2019. p. 149–155.

SCHILLING, Melissa A. Towards a general modular systems theory and its application to inter-firm product modularity. Academy of Management Review, v. 25:312-334., 1999. Disponível em: <https://ssrn.com/abstract=2263073>.

SCHWAB, Klaus. *The Fourth Industrial Revolution*. Geneva: World Economic Forum, 2016. ISBN 978-1-944835-01-9.

SHAW, M.; CLEMENTS, P. The golden age of software architecture. *IEEE Software*, v. 23, n. 2, p. 31–39, 2006.

SILVA, Stefan Santos Maciel. *Desenvolvimento de um Modelo para Apoio à Tomada de Decisão na Gestão dos Gastos da Marinha do Brasil*. Dissertação (Mestrado) — Universidade do Minho, Escola de Economia e Gestão, http://hdl.handle.net/1822/69135, 2020.

SOUZA, Vinícius José Silveira de. *Diagnóstico de falhas em arquiteturas baseadas em microsserviços*. Dissertação (Mestrado) — UNIFESP - ICT, 2021. Disponível em: <https://repositorio.unifesp.br/handle/11600/61903>.

SÖYLEMEZ, Mehmet; TEKINERDOGAN, Bedir; TARHAN, Ayça Kolukısa. Challenges and solution directions of microservice architectures: A systematic

literature review. *Applied Sciences*, v. 12, n. 11, 2022. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/12/11/5507>.

_____. Feature-driven characterization of microservice architectures: A survey of the state of the practice. *Applied Sciences*, v. 12, n. 9, 2022. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/12/9/4424>.

STAFFORD, Judith; CLEMENTS, Paul. Producing software architecture documentation to suit your needs. In: *2007 Working IEEE/IFIP Conference on Software Architecture (WICSA'07)*. [S.l.: s.n.], 2007. p. 33–33.

STOJKOV, Aleksandra; STOJANOV, Zeljko. Review of methods for migrating software systems to microservices architecture. *Journal of Engineering Management and Competitiveness*, Centre for Evaluation in Education and Science (CEON/CEES), v. 11, n. 2, p. 152–162, 2021. Disponível em: <https://doi.org/10.5937/jemc2102152s>.

STUTZEL, Matheus Costa. *Um framework para orquestração de recursos com enfoque na escalabilidade para aplicações na internet das coisas*. Dissertação (Mestrado) — Faculdade de Engenharia (UERJ), 2020.

TAPIA, Freddy et al. From monolithic systems to microservices: A comparative study of performance. *Applied Sciences*, v. 10, n. 17, 2020. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/10/17/5797>.

UNWIN, Antony. Why Is Data Visualization Important? What Is Important in Data Visualization? *Harvard Data Science Review*, v. 2, n. 1, jan 31 2020. Https://hdsr.mitpress.mit.edu/pub/zok97i7p.

VALE, Guilherme et al. *Designing Microservice Systems Using Patterns: An Empirical Study on Quality Trade-Offs*. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2201.03598>.

VELSEN, Lex van; GEEST, Thea van der; KLAASSEN, Rob. Testing the usability of a personalized system: comparing the use of interviews, questionnaires and thinking-aloud. In: *2007 IEEE International Professional Communication Conference*. [S.l.: s.n.], 2007. p. 1–8.

VILLAÇA, Luis; AZEVEDO, Leonardo; JR, Antônio. Construindo aplicações distribuídas com microsserviços. In: _____. [S.l.: s.n.], 2018. p. 1–40. ISBN 978-85-7669-452-6.

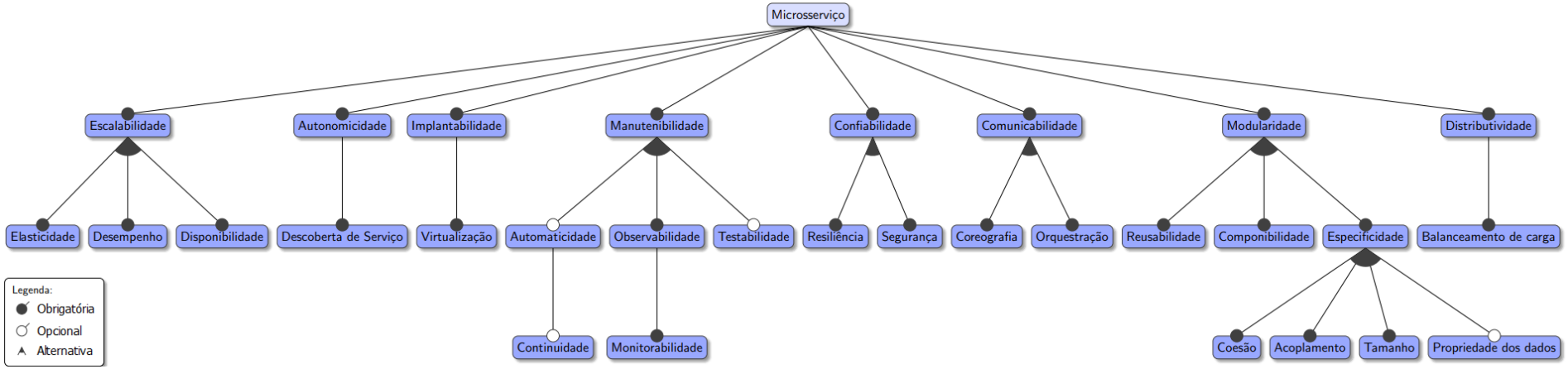
WANG, Feng-Jian; FAHMI, Faisal. Constructing a service software with microservices. In: *2018 IEEE World Congress on Services (SERVICES)*. [S.l.: s.n.], 2018. p. 43–44.

WOHLIN, Claes et al. *Experimentation in Software Engineering: An Introduction*. USA: Kluwer Academic Publishers, 2000. ISBN 0792386825.

WOLFART, Daniele et al. Modernizing legacy systems with microservices: A roadmap. In: . [S.l.: s.n.], 2021. p. 149–159.

APÊNDICE A – Modelo de Características proposto

Figura 41 - Modelo de características proposto



Fonte: O autor, 2023.

APÊNDICE B – Glossário de termos das características da arquitetura de microsserviços

Este apêndice apresenta o glossário de termos elaborado para descrever as características da arquitetura de microsserviços presentes no modelo de características.

Existe um conjunto abrangente de conceitos relacionados à AMS. Neste sentido, o presente glossário tem como objetivo apoiar equipes de desenvolvimento que estejam adotando a AMS por meio da sumarização de alguns conceitos mais relevantes, auxiliando na comunicação por meio da uniformização da terminologia de conceitos relacionados à arquitetura.

A seguir, são apresentadas as definições de cada característica e suas relações estruturais presentes no modelo de características (apresentado no Apêndice A), assim como as relações de interdependência com outras características obtidas por meio dos trabalhos pesquisados.

B.1 Escalabilidade e Características Associadas

Segundo Costa (2019), a **escalabilidade** é uma propriedade que consiste na capacidade de um sistema lidar com uma quantidade crescente de trabalho por meio da adição de recursos ao sistema. Um microsserviço escalável é aquele que consegue tratar um grande número de tarefas ou solicitações ao mesmo tempo (FOWLER, 2017), garantindo que um microsserviço funcione corretamente sem penalizar o desempenho (COJOCARU; OPRESCU; UTA, 2019). É importante observar que a escalabilidade é essencial para assegurar a disponibilidade do microsserviço (FOWLER, 2017).

A escalabilidade pode ser considerada basicamente sob dois pontos de vista: na escalabilidade horizontal, adiciona-se servidores com microsserviços implantados à estrutura de um sistema existente com o intuito de distribuir a carga de trabalho entre eles; já na escalabilidade vertical, há o aumento de recursos de *hardware* de um servidor (*i.e.*, processador, memória etc.) no qual o microsserviço está sendo executado. De acordo com Fowler (2017), o escalonamento vertical, embora seja mais fácil de se obter, não é uma forma sustentável de projetar microsserviço, pois existe a limitação inerente ao *hardware* associada à aquisição e à configuração dos dispositivos.

A Tabela 11 apresenta as interdependências relacionadas à característica escalabilidade.

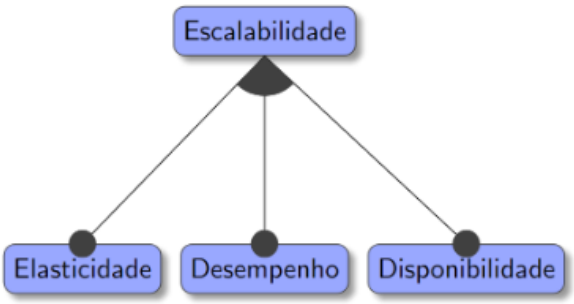
Tabela 11 - Características interdependentes à escalabilidade.

Citação	**Implicação derivada**
"*each microservice can be easily scaled in a horizontal way for better performance*"(FAHMI; HUANG; WANG, 2020)	Desempenho
"*a replicação de instâncias de microsserviços é utilizada em conjunto com balanceador de cargas e escalabilidade horizontal automática. Ao combinar as técnicas, a motivação também se amplia, uma vez que ao distribuir a carga entre as réplicas, aumentando o número réplicas caso o tráfego aumente, melhora-se o desempenho da aplicação*"(SOUZA, 2021)	Desempenho
"*a escalabilidade é essencial para a disponibilidade*"(FOWLER, 2017)	Disponibilidade
"*One approach to tackle the scalability problem was creating clusters of servers to increase the processing capacity and the availability of the service*"(BARBOSA, 2020)	Disponibilidade
"*um microsserviço que não consegue escalar experimenta uma crescente latência, baixa disponibilidade e, em casos extremos, um aumento drástico no número de incidentes e interrupções*"(FOWLER, 2017)	Desempenho, Disponibilidade, Confiabilidade e Resiliência
"*a escalabilidade automática horizontal também contribui com a tolerância à falhas. Tendo um determinado número de réplicas definido, o ambiente (runtime) mantém exatamente este número de réplicas em execução. Assim, em caso de perda de um nó ou de um serviço, um novo é automaticamente recriado para substituí-lo, tolerando a falha por recuperação*"(SOUZA, 2021)	Resiliência

Fonte: O autor, 2023.

A Figura 42 apresenta a organização das características associadas à escalabilidade, que serão detalhadas a seguir.

Figura 42 - Características associadas à escalabilidade.



Fonte: O autor, 2023.

A **elasticidade** permite o redimensionamento de recursos (*e.g.*, processamento ou armazenamento) sob demanda. A facilidade na replicação de microsserviços individuais favorece a elasticidade, sendo possível escalar dinamicamente serviços individuais conforme a necessidade. É considerada uma propriedade fundamental para a viabilização da computação em nuvem (de Sousa Neto, 2015).

A Tabela 12 apresenta a interdependência relacionada à característica elasticidade.

Tabela 12 - Característica interdependente à elasticidade.

Citação	**Implicação derivada**
"*Devido aos microsserviços poderem ser replicados e terem elasticidade, a arquitetura final possuirá uma alta disponibilidade dos serviços, uma vez que os microsserviços são auto gerenciáveis conforme a demanda de uso*"(BECKER, 2019)	Disponibilidade
"*The ease of deployment of microservices is a very desirable quality attribute for industry, usually relying on cloud infrastructure due to its automatic elasticity and on-demand resource provisioning*"(COJOCARU; OPRESCU; UTA, 2019)	Implantabilidade

Fonte: O autor, 2023.

O **desempenho** refere-se à capacidade de resposta e ao uso de recursos de um sistema (LENARDUZZI et al., 2020). Como explica Cojocaru, Oprescu e Uta (2019), o desempenho pode ser auferido em tempo de execução correlacionando o tempo de resposta (*i.e.*, o tempo entre a solicitação e a execução de um comando) e a taxa de transferência (*i.e.*, da quantidade de solicitações que podem ser processadas por unidade de tempo). Conforme Vale et al. (2022), um obstáculo para o desempenho é a latência em solicitações de rede envolvidas na comunicação entre serviços. Esse obstáculo pode ser mitigado com

a utilização de mecanismos leves de comunicação (*e.g.*, mecanismos baseados em *Representational State Transfer (REST)*, um estilo arquitetural, que fornece diretrizes para que sistemas distribuídos se comuniquem por meio de princípios e protocolos existentes na *web* sem a necessidade de protocolos mais sofisticados (NEWMAN, 2015)).

A Tabela 13 apresenta as interdependências relacionadas à característica desempenho.

Tabela 13 - Características interdependentes ao desempenho.

Citação	**Implicação derivada**
"*it is known that attribute specific analyses are interreliant, e.g., performance affects usability or availability is related to reliability*"(BOGADO; GONNET; LEONE, 2017)	Disponibilidade
"*o atributo de qualidade desempenho pode afetar os níveis de testabilidade do sistema. Uma forma de melhorar o desempenho do sistema é diminuir os níveis de indireção usados na comunicação entre dois elementos*"(PONTES, 2012)	Testabilidade

Fonte: O autor, 2023.

A **disponibilidade** é um atributo de qualidade que corresponde à proporção de tempo em que o microsserviço é acessível dentro de um período de tempo necessário. Uma pequena diminuição na disponibilidade de um microsserviço ao qual o sistema esteja altamente acoplado tende a afetar drasticamente a disponibilidade geral do sistema (COJOCARU; OPRESCU; UTA, 2019). De acordo com Becker (2019), a alta disponibilidade de uma AMS está relacionada à capacidade de replicação dos microsserviços favorecendo a elasticidade.

A Tabela 14 apresenta as interdependências relacionadas à característica disponibilidade.

Tabela 14 - Características interdependentes à disponibilidade.

Citação	**Implicação derivada**
"*Cultura de automação: Devido à complexidade que os microsserviços podem adicionar ao aumentar o número de serviços independentes do sistema, sugere-se que a empresa adote a cultura de automação, utilizando processos e ferramentas de entrega contínua, por exemplo*"(BECKER, 2019)	Automaticidade

continuação na próxima página

"*Both performance and availability can be increased if the number of servers is increased. However, this architectural decision can negatively impact the security of the system, because increasing number of servers, increases potential points of attack and failures*"(ROY; GRAHAM, 2008)	Desempenho, Segurança e Resiliência
"*availability affects safety*"(BABAR; ZHU; JEFFERY, 2004)	Segurança

Fonte: O autor, 2023.

B.2 **Autonomicidade e Característica Associada**

A **autonomicidade** é a capacidade de um microsserviço operar sem depender de outros microsserviços. Segundo Newman (2015), Araujo (2019), Tapia et al. (2020), microsserviços são pequenos serviços autônomos que trabalham em conjunto. Cada microsserviço executa em seu próprio processo e pode ter seu próprio banco de dados (VILLAÇA; AZEVEDO; JR, 2018). De acordo com Asseldonk (2021), a independência entre microsserviços é alcançada seguindo os princípios de baixo acoplamento e de alta coesão. A independência dos microsserviços contribui para a tolerância a falhas e para a disponibilidade (TAPIA et al., 2020).

A Tabela 15 apresenta as interdependências relacionadas à característica autonomicidade.

Tabela 15 - Características interdependentes à autonomicidade.

Citação	**Implicação derivada**
"*Implantação independente: Esse princípio estabelece que qualquer microsserviço deve poder ser lançado em produção sem afetar ou ter que implantar outros microsserviços*"(BECKER, 2019)	Implantabilidade
"*Independence: this characteristic emphasizes that a microservice is operationally independent of others, and the only form of communication between microservices is through their published interfaces. This is fundamental since it allows one to modify or scale a microservice without affecting the system as a whole*"(FAHMI; HUANG; WANG, 2020)	Manutenibilidade e Escalabilidade

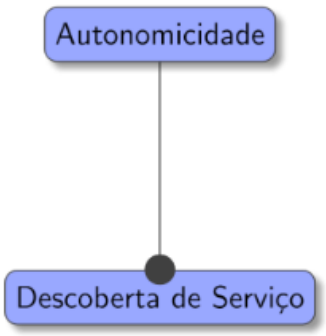
continuação na próxima página

"*To preserve independence, each microservice has its own data storage (e.g., database)*"(FAHMI; HUANG; WANG, 2020)	Propriedade dos dados
"*Cada um desses serviços implementa funções específicas, ..., e possui um alto grau de autonomia (seu próprio banco de dados) para permitir sua evolução independente dos demais*"(VILLAÇA; AZEVEDO; JR, 2018)	Propriedade dos dados e Manutenibilidade
"*a tolerância a falhas é beneficiada devido ao uso de containers e processos independentes*"(BECKER, 2019)	Resiliência
"*The independence of each one (microservices) allows it to be tolerant of failures and increases its availability*"(TAPIA et al., 2020)	Resiliência, Disponibilidade

Fonte: O autor, 2023.

A Figura 43 apresenta a organização da característica associada à autonomicidade, que será detalhada a seguir.

Figura 43 - Autonomicidade e característica associada.



Fonte: O autor, 2023.

A **descoberta de serviço** torna-se um componente essencial em um ambiente no qual instâncias de serviços são destruídas e implantadas constantemente(NEWMAN, 2022). A possibilidade de replicação e de realocação de microsserviços em tempo de execução afeta a localização estática de microsserviços em tempo de projeto, sendo necessário mecanismos para a descoberta de serviços (ARAUJO, 2019). Conforme Lehtola (2020), o registro de serviço é usado para armazenar os endereços das instâncias dos serviços para que os serviços possam localizar uns aos outros, enquanto um balanceador de carga pode direcionar o tráfego para as instâncias de serviço corretas.

A Tabela 16 apresenta as interdependências relacionadas à característica descoberta de serviço.

Tabela 16 - Características interdependentes à descoberta de serviço.

Citação	**Implicação derivada**
"*As partes da camada de comunicação relevantes para a estabilidade e a confiabilidade, além da própria rede, são a descoberta de serviços, o registro de serviços e o balanceamento de carga*"(FOWLER, 2017)	Confiabilidade
"*Service registry, when introduced, is a vital component of the system since the communication between different parts of the system depends on its availability*"(BALALAIE et al., 2018)	Disponibilidade
"*Implementing service discovery is directly dependent on system size and selected design so the most important criteria for the implementation are high availability and scalability*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Disponibilidade e Escalabilidade

Fonte: O autor, 2023.

B.3 **Implantabilidade e Característica Associada**

A **implantabilidade** é o atributo que mede o quão fácil é a adoção de um *software* por uma organização (BITTES; SANTORO; BORGES, 2005). Segundo Humble e Farley (2010), a implantação refere-se ao ato de instalar a versão de um *software* em um determinado ambiente. Em uma AMS é possível modificar um único serviço e implantá-lo de forma independente e automatizada sem impactar em outros serviços (NEWMAN, 2015), sem a necessidade de reiniciar um sistema inteiro (KAZANAVIČIUS; MAŽEIKA, 2019). Conforme Becker (2019), a implantação independente, além de aumentar a velocidade de lançamento de novos recursos, aumenta a autonomia das equipes de desenvolvimento.

A Tabela 17 apresenta as interdependências relacionadas à característica implantabilidade.

Tabela 17 - Características interdependentes à implantabilidade.

Citação	**Implicação derivada**
"*a maioria das interrupções em sistemas de produção de grande escala é causada por implantações ruins*"(FOWLER, 2017)	Confiabilidade

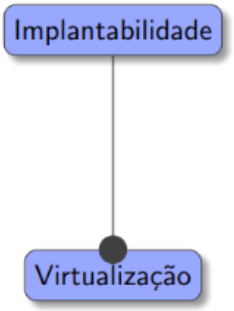
continuação na próxima página

"*há vários casos em que impedir que um MS em particular seja implantado pode aumentar a disponibilidade no ecossistema*"(FOWLER, 2017)	Disponibilidade
"*From a meaningfulness perspective, the availability is more related to the chosen deployment platform, rather than the microservice as a component*"(COJOCARU; OPRESCU; UTA, 2019)	Disponibilidade
"*the deployment characteristics make microservices a great match for the elasticity of the cloud*"(KAZANAVIČIUS; MAŽEIKA, 2019)	Elasticidade
"*The deployment quality attribute is also linked to lowering the execution cost via shortening the development cycles and increasing the microservice maintainability*"(COJOCARU; OPRESCU; UTA, 2019)	Manutenibilidade
"*A poor deployment choice can increase cost and can hurt performance, scalability, and fault tolerance*"(ABDELFATTAH; CERNY, 2023)	Desempenho, Escalabilidade e Resiliência

Fonte: O autor, 2023.

A Figura 44 apresenta a organização da característica associada à implantabilidade, que será detalhada a seguir.

Figura 44 - Implantabilidade e característica associada.



Fonte: O autor, 2023.

A **virtualização** pode ser entendida como o particionamento de um servidor físico em vários servidores lógicos (de Sousa Neto, 2015). Uma camada de abstração entre o *hardware* e o *software*, conhecida como *hypervisor*, gerencia a comunicação de cada máquina virtual com o sistema operacional hospedeiro. Além dos *softwares* e bibliotecas a serem utilizados pelo microsserviço implantado, cada máquina virtual requer seu próprio sistema operacional convidado (VILLAÇA; AZEVEDO; JR, 2018). Entretanto, em cenários com múltiplas máquinas virtuais por *host*, onde ocorra um aumento desse número de máquinas, maior será o gasto de recursos (*i.e.*, armazenamento e processamento) apenas com sistemas operacionais de cada máquina virtual (BRILHANTE, 2018).

Segundo Brilhante (2018), o desperdício de recursos na abordagem de virtualização tradicional deu origem à virtualização baseada em contêineres, como uma nova solução para tal problema. A **conteinerização** é o processo de criar, empacotar, distribuir, implantar e executar aplicações como unidades independentes em um processo de execução de ambiente leve e padronizado, conhecido como contêiner (BERG; SIEGEL; CRAMP, 2016), sendo uma alternativa mais leve em comparação ao gerenciamento de microsserviços utilizando máquinas virtuais, já que não existe a necessidade da camada intermediária, presente na virtualização (LEHTOLA, 2020). Com isso, os contêineres acessam diretamente o sistema operacional hospedeiro e seus recursos para prover um ambiente virtual para as aplicações. A separação causada pelo isolamento a nível de disco, memória, processamento e rede permite grande flexibilidade, onde ambientes distintos podem coexistir, sem problemas, na mesma máquina hospedeira. Assim, no ambiente de contêiner, é necessário instalar as dependências do microsserviço (*i.e.*, *softwares*, arquivos etc.) sem se preocupar com a instalação de outro sistema operacional. (VILLAÇA; AZEVEDO; JR, 2018).

A Tabela 18 apresenta a interdependência relacionada à característica virtualização.

Tabela 18 - Característica interdependente à virtualização.

Citação	**Implicação derivada**
"*a virtualização e a computação em nuvem facilitaram bastante o provisionamento da infraestrutura mas, ao mesmo tempo, a quantidade de elementos a ser monitorada aumentou significamente*"(LEVITA, 2017)	Monitorabilidade
"*Orquestradores de contêineres como o Kubernetes suportam escalabilidade horizontal baseada em métricas de uso de recursos, como CPU e memória*"(SOUZA, 2021)	Escalabilidade
"*a tolerância a falhas é beneficiada devido ao uso de containers e processos independentes*"(BECKER, 2019)	Resiliência
"*Containers better support the microservice design philosophies of small stateless services, rapid and automated scalability, speed to change, and distributed deployment*"(BERG; SIEGEL; CRAMP, 2016)	Tamanho, Escalabilidade, Manutenibilidade e Implantabilidade
"*Since containers running on the same host share the services of the one Linux kernel, there is a greater security risk present in comparison to running the same applications within their own dedicated virtual machines*"(BERG; SIEGEL; CRAMP, 2016)	Segurança

Fonte: O autor, 2023.

B.4 **Manutenibilidade e Características Associadas**

A **manutenibilidade** é o equilíbrio entre o tempo, os recursos necessários para aplicar mudanças e a necessidade de introduzir novas lógicas ou melhorar funcionalidades de um sistema (VALE et al., 2022). Ela pode ser compreendida como o grau de eficácia e eficiência com que um produto ou sistema pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo às mudanças no ambiente e nos requisitos (SANTOS, 2020). Serviços pequenos e autônomos são mais fáceis de entender, desenvolver e manter (VILLAÇA; AZEVEDO; JR, 2018; FAHMI; HUANG; WANG, 2020). De acordo com Lehtola (2020), gerenciar cada serviço de forma independente ajuda a reduzir a complexidade e, por consequência, aumenta a manutenibilidade.

A Tabela 19 apresenta a interdependência relacionada à característica manutenibilidade.

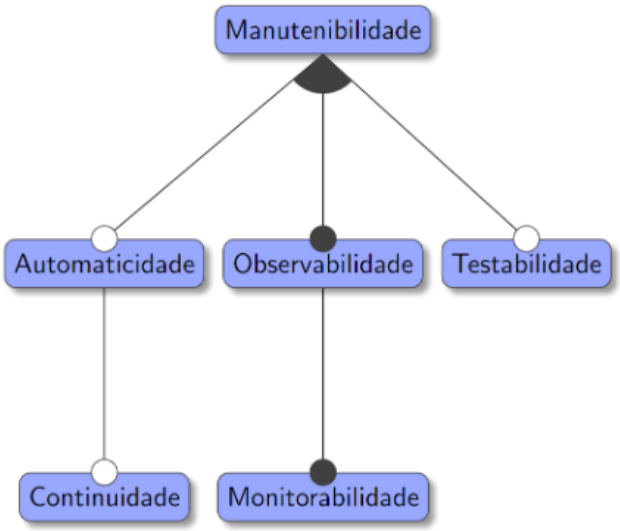
Tabela 19 - Característica interdependente à manutenibilidade.

Citação	**Implicação derivada**
"*MSA provides maintainability to the system and increases the rate at which software products are delivered*"(COJOCARU; OPRESCU; UTA, 2019)	Automaticidade
"*a microservice is considered maintainable if it presents high degree of cohesion, low degree of coupling and homogeneous technology stack*"(COJOCARU; OPRESCU; UTA, 2019)	Coesão e Acoplamento
"*Fine granularity, decreased level of system's technology heterogeneity and good scalability indicate good maintainability*"(COJOCARU; OPRESCU; UTA, 2019)	Tamanho e Escalabilidade

Fonte: O autor, 2023.

A Figura 45 apresenta a organização das características associadas à manutenibilidade, que serão detalhadas a seguir.

Figura 45 - Características associadas à manutenibilidade.



Fonte: O autor, 2023.

A **automaticidade** é uma propriedade que pode ser compreendida como a capacidade de automatização de procedimentos relacionados à entrega de *software*. Segundo Humble e Farley (2010), se o processo completo de entrega de *software* não é automatizado, ele não é passível de repetição. Como os passos são manuais, são passíveis de erro e não existe uma forma de rever exatamente o que foi feito. Com isso, não há como ter controle sobre o processo de entrega. Conforme Newman (2022), a automação pode trazer benefícios, como o aumento da velocidade das entregas, a facilidade na detecção

de incidentes de segurança, a redução da variabilidade de resultados e também de erros humanos. Segundo Söylemez, Tekinerdogan e Tarhan (2022a), Lehtola (2020), a automação da infraestrutura é essencial para obter a maioria dos benefícios dos microsserviços e está diretamente relacionada a *pipelines* utilizando a integração contínua (*continuous integration*) e a entrega contínua (*continuous delivery*).

A Tabela 20 apresenta as interdependências relacionadas à característica automaticidade.

Tabela 20 - Características interdependentes à automaticidade.

Citação	**Implicação derivada**
"*introduzir um pipeline de deployment padronizado pode ajudar a garantir a estabilidade e a confiabilidade em todo o ecossistema de microsserviços*"(FOWLER, 2017)	Confiabilidade
"*Automated CI/CD pipeline is essential for gaining most of the benefits of microservices such as fast deployment*"(LEHTOLA, 2020)	Implantabilidade
"*a automação pode nos ajudar na recuperação logo após um incidente. Podemos usá-la para revogar e fazer uma rotação das chaves de segurança, além de utilizar ferramentas que ajudem a detectar possíveis problemas de segurança com mais facilidade. Assim como em outros aspectos da arquitetura de microsserviços, adotar uma cultura de automação ajudará bastante quando o assunto é a segurança*"(NEWMAN, 2022)	Segurança

Fonte: O autor, 2023.

A **continuidade**, refere-se à utilização das práticas de integração contínua (IC) e de entrega contínua (EC). A IC, conforme Beck (1999), é uma prática usada na engenharia de *software* que consiste na integração e teste do código desenvolvido pelo menos uma vez ao dia. Conforme Humble e Farley (2010), a integração das alterações no sistema de controle de versão deve ser frequente. Segundo O'Connor, Elger e Clarke (2017), a IC favorece alguns benefícios para as organizações de desenvolvimento de *software*, como a melhora na velocidade de entrega de novas funcionalidades e a redução de riscos de integração, já que erros são encontrados mais cedo.

A EC, segundo Humble e Farley (2010), é uma prática que tem como objetivo entregar versões novas e funcionais de um sistema várias vezes ao dia. A IC é um pré-requisito para a EC (KIM et al., 2016). Segundo Lehtola (2020), a EC é uma prática, frequentemente associada a microsserviços, em que o *software* é sempre mantido pronto para liberação. A ideia é garantir que o código, que passou nos testes automatizados,

esteja pronto para ser implantado em um determinado ambiente de forma rápida e segura (HUMBLE; FARLEY, 2010). Conforme O'Connor, Elger e Clarke (2017), toda vez que uma alteração é confirmada no repositório de controle de versão, o código é compilado, se necessário, empacotado por um servidor de compilação para então ser testado por diversas diferentes técnicas antes que possa ter seu *status* classificado como "liberado".

A Tabela 21 apresenta as interdependências relacionadas à característica continuidade.

Tabela 21 - Características interdependentes à continuidade.

Citação	**Implicação derivada**
"*continuous integration, a set of software engineering practices that speed up the delivery of software by decreasing integration times*"(O'CONNOR; ELGER; CLARKE, 2017)	Implantabilidade
"*Establishing continuous integration has been mentioned as one of the firsts steps when deploying microservices*"(LEHTOLA, 2020)	Implantabilidade
"*the adoption of DevOps practices like continuous integration (CI) and continuous delivery (CD) will help make the deployment process more effective*"(LEHTOLA, 2020)	Implantabilidade
"*Continuous integration (CI) and continuous delivery (CD) are DevOps practices that make the handling, deployment, and testing of microservices easier and more efficient*"(O'CONNOR; ELGER; CLARKE, 2017)	Implantabilidade e Testabilidade

Fonte: O autor, 2023.

A **observabilidade** é uma propriedade de sistema que consiste na capacidade que um sistema possui em permitir a compreensão de seu estado interno a partir de suas saídas. Quanto mais observável é um sistema, mais fácil será o entendimento de um problema quando algo der errado (NEWMAN, 2022).

A Tabela 22 apresenta a interdependência relacionada à característica observabilidade.

Tabela 22 - Característica interdependente à observabilidade.

Citação	**Implicação derivada**

continuação na próxima página

"*Uma vez que as aplicações sejam observáveis, o monitoramento pode ser feito, possibilitado que análises sejam realizadas, alertas sejam gerados e dashboards atualizados*"(SOUZA, 2021)	Monitorabilidade

Fonte: O autor, 2023.

A **monitorabilidade** é a capacidade de um sistema fornecer informações obtidas por meio do monitoramento de um sistema em tempo de execução (VALE et al., 2022). De acordo com Becker (2019), os benefícios fornecidos pela divisão de um sistema em serviços com granularidade fina são proporcionais à complexidade do monitoramento desses serviços. Nesse cenário complexo, haverá a necessidade de ferramentas de gerenciamento e monitoramento automatizadas para lidar com a natureza heterogênea de vários serviços. Um monitoramento adequado contribui para a disponibilidade de um microsserviço (FOWLER, 2017).

A Tabela 23 apresenta as interdependências relacionadas à característica monitorabilidade.

Tabela 23 - Características interdependentes à monitorabilidade.

Citação	**Implicação derivada**
"*Aplicações com monitoramento adequado facilitam o diagnóstico de falhas e correção, contribuindo com a confiabilidade e disponibilidade geral dos sistemas*"(SOUZA, 2021)	Confiabilidade e Disponibilidade
"*outro princípio necessário para garantir a disponibilidade de um MS é seu adequado monitoramento*"(FOWLER, 2017)	Disponibilidade
"*Monitoring, tracing, and logging are important activities to ensure that systems are able to satisfy availability, performance, and reliability concerns*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Disponibilidade, Desempenho e Confiabilidade
"*This concept of fault tolerance leads to a new characteristic of microservices oriented architectures, the pressing need to constantly monitor the execution of each service in a different way and the containment measures that must be carried out to guarantee the availability of the business*"(BARBOSA, 2020)	Resiliência e Disponibilidade

Fonte: O autor, 2023.

A **testabilidade** é a capacidade de um sistema demonstrar suas falhas por meio

de testes (VALE et al., 2022). Consiste no grau de eficácia e eficiência com o qual os critérios de teste podem ser estabelecidos para um sistema, produto ou componente e testes podem ser realizados para determinar se esses critérios foram atendidos (SANTOS, 2020). Os desafios relacionados a testar funcionalidades envolvendo sistemas distribuídos continuam presentes (NEWMAN, 2022). De acordo com Costa (2019), ao analisar a literatura sobre a AMS, problemas relacionados à baixa testabilidade nessa arquitetura são encontrados. Devido à complexa relação entre os microsserviços, a testabilidade é uma propriedade importante que ajuda a detectar e a minimizar impactos em outros atributos como desempenho, segurança ou disponibilidade (VALE et al., 2022).

A Tabela 24 apresenta as interdependências relacionadas à característica testabilidade.

Tabela 24 - Características interdependentes à testabilidade.

Citação	**Implicação derivada**
"*Due to the complex relationship between services, testability is an important property that helps to detect and minimize impacts on other qualities like performance, security or availability*"(VALE et al., 2022)	Desempenho, Segurança e Disponibilidade

Fonte: O autor, 2023.

B.5 **Confiabilidade e Características Associadas**

A **confiabilidade** consiste no grau em que um sistema, produto ou componente de *software* desempenha sua função sem falhas durante um determinado período de tempo (SANTOS; WERNER, 2019; SOUZA, 2021). Assim, quanto maior for o tempo em funcionamento sem apresentar falhas, maior será o nível de confiabilidade de um serviço.

A Tabela 25 apresenta as interdependências relacionadas à característica confiabilidade.

Tabela 25 - Características interdependentes à confiabilidade.

Citação	**Implicação derivada**
"*o roteamento e a descoberta de serviços são requisitos da confiabilidade*"(FOWLER, 2017)	Descoberta de Serviços
"*Ao agregar confiabilidade aos microsserviços, protege-se sua disponibilidade*"(FOWLER, 2017)	Disponibilidade

continuação na próxima página

"*a disponibilidade requer que a comunicação e o roteamento entre diferentes serviços sejam confiáveis*"(FOWLER, 2017)	Disponibilidade

Fonte: O autor, 2023.

A Figura 46 apresenta a organização das características associadas à confiabilidade, que serão detalhadas a seguir.

Figura 46 - Características associadas à confiabilidade.

Fonte: O autor, 2023.

A **resiliência** pode ser compreendida como o grau em que um sistema, produto ou componente opera conforme pretendido, apesar da presença de falhas de *hardware* ou *software* (SANTOS, 2020). Segundo Araujo (2019), a independência natural dos serviços em uma AMS permite isolar os principais serviços em sua própria infraestrutura com o objetivo de mantê-los funcionando, mesmo que falhas sejam ocasionadas por outros serviços. Uma falha em cascata em todo o sistema deveria ser impossível (LEHTOLA, 2020). Conforme Becker (2019), a utilização de *containers* e de processos independentes melhora o isolamento e, por consequência, aumenta a tolerância a falhas. Essa independência favorece também ao aumento da disponibilidade (TAPIA et al., 2020).

A Tabela 26 apresenta as interdependências relacionadas à característica resiliência.

Tabela 26 - Características interdependentes à resiliência.

Citação	**Implicação derivada**
"*a tolerância a falhas é beneficiada devido ao uso de containers e processos independentes, pois um único microsserviço é completamente isolado de outros microsserviços e só pode ser afetado por eles através das suas interfaces ou através dos recursos dos quais que ele depende, ou seja, mesmo que um microsserviço venha a falhar, outros serviços não serão necessariamente afetados*"(BECKER, 2019)	Virtualização e Autonomicidade
"*resiliência e tolerância à falhas são requisitos essenciais a um sistema baseado em microsserviços, visando evitar a indisponibilidade*"(SOUZA, 2021)	Disponibilidade

Fonte: O autor, 2023.

A **segurança** consiste no grau em que um produto ou sistema protege informações e dados para que pessoas ou outros produtos ou sistemas tenham o grau de acesso a dados adequado a seus tipos e níveis de autorização (SANTOS, 2020). Na AMS, como há mais dados trafegando entre os diversos serviços por meio de redes, a expansão da superfície de ataque é maior. Quanto maior a comunicação e a quantidade de aplicações interativas, maior serão as chances de ataques. Por outro lado, os microsserviços possibilitam uma defesa em profundidade, por meio da utilização de padrões de autenticação e acesso, podendo reduzir o impacto de um ataque caso ele ocorra (NEWMAN, 2022).

A Tabela 27 apresenta a interdependência relacionada à característica segurança.

Tabela 27 - Característica interdependente à segurança.

Citação	**Implicação derivada**
"*o processo de verificação (além de todos os elementos e abstrações do sistema relacionados à segurança: unidade certificadora, unidade verificadora, listas de controle de acesso, entre outros) deteriorará o desempenho da aplicação, dado que consumirá recursos que poderiam ser destinados à operação em si*"(PONTES, 2012)	Desempenho

Fonte: O autor, 2023.

B.6 **Comunicabilidade e Características Associadas**

Define-se **comunicabilidade** como a facilidade observada na comunicação entre microsserviços, de forma que a confiabilidade seja preservada. Os microsserviços precisam ser capazes de se descobrir e de se comunicar de forma consistente sem perda de confiança no processo (LEHTOLA, 2020). Com muitos serviços pequenos interagindo, a comunicação entre os microsserviços deve ser eficiente e robusta (MICROSOFT, 2022). No entanto, como a comunicação se dá por meio da rede, há uma propensão a falhas ou à lentidão nesse processo (TAPIA et al., 2020).

A Tabela 28 apresenta as interdependências relacionadas à característica comunicabilidade.

Tabela 28 - Características interdependentes à comunicabilidade.

Citação	**Implicação derivada**
"*Communication and integration are other challenging points that have emerged as a result of distributed architecture. Even if microservices communicate with a more lightweight protocol, it is still difficult to ensure that the communication infrastructure is reliable and the protocol to be used for communication and integration can handle complex workflows. The most important criteria for both challenges are reliability and durability; if these criteria are not met, the proper operation and reliability of the system will be affected, and will possibly cause cascading failures in the system*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Confiabilidade e Resiliência
"*the latency in network requests involved in inter-service communication is an obvious source of performance hindrance. Established lightweight and REST-based communication mechanisms are commonly used to mitigate this threat*"(VALE et al., 2022)	Desempenho
"*Em casos de aplicações cujos requisitos sejam de maior de desempenho e menor tempo de resposta, um protocolo comumente utilizado é o gRPC sobre o HTTP 2.0. Para obter um desempenho melhor que o HTTP usual, o gRPC transmite dados em forma binária, codificando-os no formato de Protocol Buffers*"(SOUZA, 2021)	Desempenho

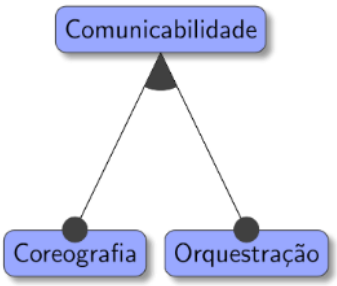
continuação na próxima página

"*a sua topologia de comunicação entre processos, fundamentada em ambientes distribuídos, também acarreta desafios relacionados à disponibilidade dos serviços*"(SOUZA, 2021)	Disponibilidade
"*Por ser assíncrona, a comunicação orientada a mensagens contribui para o aumento da disponibilidade do sistema. Isso ocorre pois não é necessário que os receptores estejam em execução para que as mensagens sejam enviadas, desbloqueando a continuidade da execução dos emissores de requisição*"(SOUZA, 2021)	Disponibilidade
"*Communication in a microservices architecture is one of the most challenging points due to its distributed nature. It directly affects the availability and resiliency of the system*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022b)	Disponibilidade e Resiliência
"*Quanto maior a comunicação e a quantidade de aplicações interativas, maior serão as chances de ataque*"(NEWMAN, 2022)	Segurança

Fonte: O autor, 2023.

A Figura 47 apresenta a organização das características associadas à comunicabilidade, que serão detalhadas a seguir.

Figura 47 - Características associadas à comunicabilidade.



Fonte: O autor, 2023.

A **coreografia** refere-se a uma abordagem de composição de serviços no qual cada serviço envolvido sabe exatamente quando executar suas operações e com quem interagir, sem que haja um controle centralizado. As coreografias podem ser vistas como um acordo entre um conjunto de serviços sobre como uma determinada colaboração deve ocorrer (ARAUJO, 2019). Conforme Fahmi, Huang e Wang (2020), os benefícios dessa abordagem são o baixo acoplamento e a maior agilidade, e sua desvantagem é a dificuldade de manter

e de gerenciar os processos de negócios. Devido ao menor acoplamento, pela inexistência do orquestrador, e pelo favorecimento ao escalonamento e à manutenibilidade, a AMS tende a se adaptar melhor com a composição por meio de coreografia (NEWMAN, 2015).

A Tabela 29 apresenta as interdependências relacionadas à característica coreografia.

Tabela 29 - Características interdependentes à coreografia.

Citação	**Implicação derivada**
"*The benefits of choreography approach are loose-couple and more agile, and its drawback is difficult to maintain and manage the business process*"(FAHMI; HUANG; WANG, 2020)	Acoplamento e Manutenibilidade

Fonte: O autor, 2023.

A **orquestração**, de acordo com Araujo (2019), refere-se a uma abordagem para a composição de serviços no qual os serviços são controlados de maneira centralizada. Segundo Fahmi, Huang e Wang (2020), os benefícios dessa abordagem são a facilidade de manter e de gerenciar os processos de negócios e sua desvantagem é o alto acoplamento entre o orquestrador e os microsserviços.

A Tabela 30 apresenta as interdependências relacionadas à característica confiabilidade.

Tabela 30 - Características interdependentes à orquestração.

Citação	**Implicação derivada**
"*The benefits of orchestration approach are easier to maintain and manage the business process and its drawback is tight-couple between the orchestrator and microservices"*(FAHMI; HUANG; WANG, 2020)	Manutenibilidade e Acoplamento

Fonte: O autor, 2023.

B.7 **Modularidade e Características Associadas**

A **modularidade**, um conceito geral de sistemas, pode ser compreendido como o grau em que os componentes de um sistema podem ser separados e recombinados (SCHILLING, 1999). É o grau em que um sistema ou programa de computador é composto de componentes discretos, de forma que uma alteração em um componente tenha um impacto mínimo em outros componentes (SANTOS, 2020). De acordo com Cojocaru, Oprescu e Uta (2019), é um atributo de qualidade baseado no princípio de responsabilidade única,

que também é influenciado pela complexidade do sistema, levando-se em consideração a granularidade do microsserviço e a separação em partes específicas de negócio. Um bom nível de modularidade é garantido, na prática, pela boa definição da fronteira de um microsserviço (VILLAÇA; AZEVEDO; JR, 2018).

A Tabela 31 apresenta as interdependências relacionadas à característica modularidade.

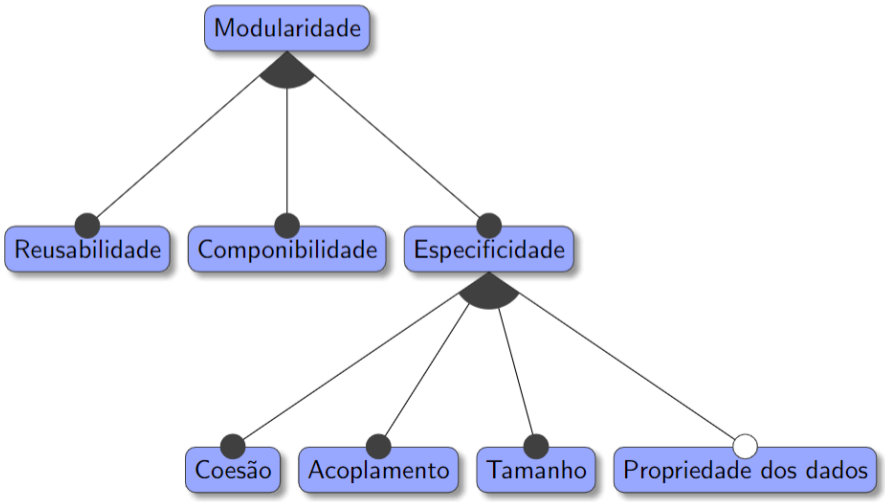
Tabela 31 - Características interdependentes à modularidade.

Citação	**Implicação derivada**
"*a quality attribute relying mostly on cohesion through the rule of single responsibility. It is also influenced by the complexity of the system, referring to the granularity of the microservice and the separation of business concerns*"(COJOCARU; OPRESCU; UTA, 2019)	Coesão e Tamanho
"*Good modularity allows me to only have to understand a small subset of the code base to make a change*"(FOWLER, 2019)	Manutenibilidade
"*Some of the benefits of product modularization include reduction of fixed costs of developing individual product variants, greater degree of components and subsystems reuse, increased responsiveness, offer higher product variety to customers, reduction of development lead time, and improved customer service*"(HANSEN; SUN, 2011)	Reusabilidade

Fonte: O autor, 2023.

A Figura 48 apresenta a organização das características associadas à modularidade, que serão detalhadas a seguir.

Figura 48 - Características associadas à modularidade.



Fonte: O autor, 2023.

Segundo Krueger (1992), o reuso de *software* é o processo de criação de sistemas de *software* a partir de *software* existente em vez de construir sistemas de *software* do zero. Assim, a **reusabilidade** é a capacidade de aproveitamento de partes de um *software*. É um atributo de qualidade que possui forte relação com a perspectiva de custos, já que a manutenibilidade resultante da construção de um microsserviço com alta reusabilidade pode justificar o alto custo para o seu desenvolvimento (COJOCARU; OPRESCU; UTA, 2019). Conforme Kumar, Singh e Dawra (2022), encontrar componentes de *software* altamente coesos para reuso pode reduzir o esforço de manutenção e aumentar a velocidade no desenvolvimento de *software*.

A Tabela 32 apresenta a interdependência relacionada à característica reusabilidade.

Tabela 32 - Característica interdependente à reusabilidade.

Citação	**Implicação derivada**
"*software reusability is an emerging software engineering technique that improves the quality of software development, reduced maintainability and shorten development time*"(KUMAR; SINGH; DAWRA, 2022)	Manutenibilidade

Fonte: O autor, 2023.

A **componibilidade** é um princípio de *design* onde um ambiente é decomposto em componentes interoperáveis que podem ser implantados independentemente e reutilizados em outros ambientes (BERG; SIEGEL; CRAMP, 2016). Tal característica, uma das principais promessas dos sistemas distribuídos e das AOS, refere-se a capacidade de criação de oportunidades para a reutilização de funcionalidades (NEWMAN, 2022). Na

AMS, sistemas são compostos por serviços que colaboram entre si para atingirem seus objetivos (VILLAÇA; AZEVEDO; JR, 2018). Dessa forma, os serviços devem ser criados permitindo que funcionalidades sejam consumidas de diferentes maneiras e para diferentes propósitos (NEWMAN, 2022). De acordo com Söylemez, Tekinerdogan e Tarhan (2022a), a decomposição permite a criação de serviços autônomos, altamente coesos e fracamente acoplados.

A Tabela 33 apresenta as interdependências relacionadas à característica componibilidade.

Tabela 33 - Características interdependentes à componibilidade.

Citação	**Implicação derivada**
"*Decomposition allows us to have autonomous services organized around business capability services. it is necessary to separate the system into suitable pieces functionally and obtaining high cohesive and loosely coupled services are expected as a result of decomposition*" (SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Autonomicidade, Coesão e Acoplamento

Fonte: O autor, 2023.

A **especificidade** pode ser compreendida como o grau de responsabilidade atribuído a um microsserviço por meio de funcionalidades a serem executadas. Quanto menor o grau de responsabilidade atribuído, mais específico será o microsserviço. De acordo com Asseldonk (2021), o tamanho pequeno dos microsserviços possui relação com o princípio de responsabilidade única, contribuindo para a maior coesão do microsserviço (COSTA, 2019). Os microsserviços são construídos em torno de um domínio específico do negócio por meio de funções específicas (VILLAÇA; AZEVEDO; JR, 2018).

A Tabela 34 apresenta a interdependência relacionada à característica especificidade.

Tabela 34 - Característica interdependente à especificidade.

Citação	**Implicação derivada**
"*Cada microsserviço tem uma fronteira muito bem definida o que garante o nível de modularidade na prática, o que é muito difícil de alcançar em uma aplicação monolítica*"(VILLAÇA; AZEVEDO; JR, 2018)	Modularidade

Fonte: O autor, 2023.

A **coesão** representa o grau em que as operações fornecidas por um microsserviço atendam a apenas uma funcionalidade (AL-DEBAGY; MARTINEK, 2020; COJOCARU;

OPRESCU; UTA, 2019). Segundo Newman (2022), as funcionalidades devem ser agrupadas de modo que alterações sejam feitas no menor número possível de locais. Se for necessário alterar algum comportamento, deve ser possível fazê-lo em um só local, lançando essa alteração o mais rápido possível.

A Tabela 35 apresenta as interdependências relacionadas à característica coesão.

Tabela 35 - Características interdependentes à coesão.

Citação	**Implicação derivada**
"*Independence between microservices is achieved by following the loose coupling and high cohesion principle*"(ASSELDONK, 2021)	Autonomicidade
"*a microservice is considered maintainable if it presents high degree of cohesion, low degree of coupling*"(COJOCARU; OPRESCU; UTA, 2019)	Manutenibilidade

Fonte: O autor, 2023.

O **acoplamento**, segundo Cojocaru, Oprescu e Uta (2019), representa as dependências e conexões de um microsserviço para outro. De acordo com Newman (2022), uma mudança em um serviço com baixo acoplamento não deve exigir que outro serviço seja alterado. Segundo Lehtola (2020), como um microsserviço deve ser independente, construído em torno de uma capacidade de negócios, com foco em responsabilidade única, os limites do serviço devem ser definidos de forma que alterações em uma funcionalidade de um serviço não gerem alterações em outros serviços, ou seja, o acoplamento entre os serviços deve ser baixo.

A Tabela 36 apresenta as interdependências relacionadas à característica acoplamento.

Tabela 36 - Características interdependentes ao acoplamento.

Citação	**Implicação derivada**
"*Independence between microservices is achieved by following the loose coupling and high cohesion principle*"(ASSELDONK, 2021)	Autonomicidade
"*slight decrease in the availability of one microservice that has high coupling is dramatically decreasing the overall availability of the system*"(COJOCARU; OPRESCU; UTA, 2019)	Disponibilidade

continuação na próxima página

"*a microservice is considered maintainable if it presents high degree of cohesion, low degree of coupling*"(COJOCARU; OPRESCU; UTA, 2019)	Manutenibilidade
"*The coupling of different services may imply that changes to a service cannot be realized without orchestrating similar changes in services depending on it, which is a sign of poor maintainability*"(VALE et al., 2022)	Manutenibilidade

Fonte: O autor, 2023.

O atributo de qualidade **tamanho**, também referenciado por granularidade, é o mais controverso em relação a um microsserviço. A controvérsia decorre da falta de uma definição geralmente aceita para o tamanho ideal para um microsserviço (COJOCARU; OPRESCU; UTA, 2019). Conforme Fahmi, Huang e Wang (2020), há uma ênfase no tamanho pequeno dos microsserviços em relação aos serviços introduzidos pela AOS. De acordo com Asseldonk (2021), o pequeno tamanho traz grandes benefícios em termos de manutenibilidade e extensibilidade, contribuindo para a produtividade das equipes de desenvolvimento (VILLAÇA; AZEVEDO; JR, 2018).

A Tabela 37 apresenta as interdependências relacionadas à característica tamanho.

Tabela 37 - Características interdependentes ao tamanho.

Citação	**Implicação derivada**
"*Due to smaller in size and independence characteristics, each microservice can be easily scaled in a horizontal way for better performance*"(FAHMI; HUANG; WANG, 2020)	Escalabilidade e Desempenho
"*the size of a microservice directly impacts its performance and availability*"(KLOCK et al., 2017 apud SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Desempenho e Disponibilidade
"*The smaller brings the benefit in maintainability: A small service can be easily modified, and, if needed, to be rebuilt from the scratch with limited resources and in limited time*"(FAHMI; HUANG; WANG, 2020)	Manutenibilidade
"*The microservices have to be small-sized. The small size brings major benefits in terms of maintainability and extendability*"(ASSELDONK, 2021)	Manutenibilidade

continuação na próxima página

"*Apesar dos benefícios em quebrar um sistema em serviços com granularidade fina, isso acaba gerando uma complexidade alta no monitoramento do comportamento de múltiplos serviços*"(BECKER, 2019)	Monitorabilidade

Fonte: O autor, 2023.

A **propriedade dos dados** pode ser compreendida como a capacidade de um microsserviço armazenar e possuir todos os dados que estão relacionados a sua funcionalidade. Segundo Araujo (2019), Fahmi, Huang e Wang (2020), por serem entidades autônomas e a fim de reduzir o acoplamento, cada microsserviço pode ter seu próprio banco de dados. De acordo com Lehtola (2020), é possível que alguns serviços não utilizem um banco de dados, sendo mais comum que microsserviços utilizem diferentes bancos de dados, o que é um grande desafio para a gestão de dados. De acordo com Barbosa (2020), microsserviços relacionados a um domínio específico de negócio tendem a manter seu próprio banco de dados para atender a esse domínio.

A Tabela 38 apresenta as interdependências relacionadas à propriedade dos dados.

Tabela 38 - Características interdependentes à propriedade dos dados.

Citação	**Implicação derivada**
"*To preserve independence, each microservice has its own data storage (e.g., database)*"(FAHMI; HUANG; WANG, 2020)	Autonomicidade
"*O desempenho pode ser afetado ao executar operações que coletam dados de vários serviços (com diferentes meios de armazenamentos de dados)*"(VILLAÇA; AZEVEDO; JR, 2018)	Desempenho

Fonte: O autor, 2023.

B.8 **Distributividade e Característica Associada**

Quanto à **distributividade**, os microsserviços são uma abordagem para arquitetar sistemas de *software* distribuídos usando serviços independentes de granularidade fina (STOJKOV; STOJANOV, 2021). Eles possuem uma natureza descentralizada e autônoma, o que indica que podem estar em diferentes servidores (ARAUJO, 2019; LEHTOLA, 2020). Segundo Fahmi, Huang e Wang (2020), uma AMS pode ser composta por diversos microsserviços distribuídos, cada um executando em seu próprio processo e se comunicando com outros por meio de mecanismos baseados em mensagens.

A Tabela 39 apresenta as interdependências relacionadas à distributividade.

Tabela 39 - Características interdependentes à distributividade.

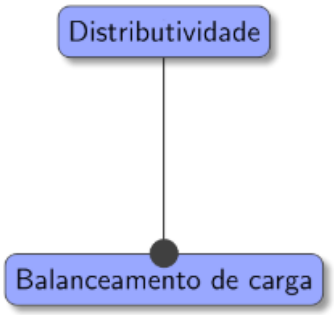
Citação	**Implicação derivada**
"*Communication in a microservices architecture is one of the most challenging points due to its distributed nature*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022b)	Comunicabilidade
"*Neste tipo de ambiente distribuído, falhas são mais prováveis. Além disso, é consenso que sistemas distribuídos sejam naturalmente mais complexos de serem analisados e compreendidos, quando comparados à sistemas monolíticos tradicionais, dificultando o diagnóstico de problemas e retardando eventuais correções*"(VILLAÇA; AZEVEDO; JR, 2018)	Confiabilidade, Observabilidade e Manutenibilidade
"*Service discovery, on the other hand, is an indispensable structure in the distributed architecture. Thanks to this structure, the changing services, whose location information is dynamic, become able to discover after they complete service-registration*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022b)	Descoberta de Serviço
"*As a distributed architecture, the latency in network requests involved in inter-service communication is an obvious source of performance hindrance*"(VALE et al., 2022)	Desempenho
"*Devido à complexidade na implementação de transações distribuídas, as arquiteturas de microsserviço enfatizam a coordenação em atualizações de serviço e delegações em consultas, de modo que a consistência ou a disponibilidade nem sempre serão garantidas*"(VILLAÇA; AZEVEDO; JR, 2018)	Disponibilidade
"*The distribution of application logic into different processes results in complex network interaction models between services. As a result, this added complexity can be more easily exploited by attackers if security is neglected*"(VALE et al., 2022)	Segurança

Fonte: O autor, 2023.

A Figura 49 apresenta a organização da característica associada à distributividade,

que será detalhada a seguir.

Figura 49 - Distributividade e característica associada.



Fonte: O autor, 2023.

O **balanceamento de carga** é a prática frequentemente empregada para dividir o tráfego de rede entre as instâncias dos serviços (LEHTOLA, 2020). Conforme Newman (2015), quando se deseja aumentar a resiliência de um serviço, deve-se evitar pontos únicos de falha. Assim, um balanceador de carga poderá ser utilizado. Os elementos da camada de comunicação considerados relevantes para a estabilidade e a confiabilidade, além da própria rede, são a descoberta de serviços, o registro de serviços e o balanceamento de carga (FOWLER, 2017). Segundo Cojocaru, Oprescu e Uta (2019), representa a distribuição da carga de trabalho entre microsserviços e só tem sentido no escalonamento horizontal.

A Tabela 40 apresenta as interdependências relacionadas ao balanceamento de carga.

Tabela 40 - Características interdependentes ao balanceamento de carga.

Citação	**Implicação derivada**
"*As partes da camada de comunicação relevantes para a estabilidade e a confiabilidade, além da própria rede, são a descoberta de serviços, o registro de serviços e o balanceamento de carga*"(FOWLER, 2017)	Confiabilidade
"*A service registry is used to store the service instances' addresses so that the services can locate each other, and a load balancer can direct traffic to the right service instances*"(LEHTOLA, 2020)	Descoberta de Serviço

continuação na próxima página

"*a replicação de instâncias de microsserviços é utilizada em conjunto com balanceador de cargas e escalabilidade horizontal automática. Ao combinar as técnicas, a motivação também se amplia, uma vez que ao distribuir a carga entre as réplicas, aumentando o número réplicas caso o tráfego aumente, melhora-se o desempenho da aplicação*"(SOUZA, 2021)	Desempenho
"*The most important criteria for load balancing are availability because requests that are not effectively distributed will cause the system to stop responding, which will affect system availability*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Disponibilidade
"*Another concern is load balancing. It is used to distribute the traffic coming to the system efficiently. It also provides high availability and reliability by sending incoming requests only to the servers that are standing*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022b)	Disponibilidade e Confiabilidade
"*load balancing is the most important mechanism for availability and scalability and there are some techniques such as client-side and server-side to implement load balancing in a system*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022a)	Disponibilidade e Escalabilidade
"*alguns balanceadores de carga como o AWS Elastic Load Balancer fornecem escalabilidade automática agendada, variando o número de réplicas em horários específicos de acordo com o tráfego e carga do sistema*"(SOUZA, 2021)	Escalabilidade
"*Para garantir a distribuição da carga de trabalho entre as réplicas, um balanceador de carga deve ser utilizado. Dessa forma, balanceadores de carga contribuem indiretamente com a prevenção e tolerância à falhas*"(SOUZA, 2021)	Resiliência
"*Client-side load balancing is often used in some scenarios as it eliminates a single point of failure*"(SÖYLEMEZ; TEKINERDOGAN; TARHAN, 2022b)	Resiliência

Fonte: O autor, 2023.

APÊNDICE C – Questionário de avaliação

Este apêndice apresenta o questionário elaborado para a avaliação do atendimento às características da arquitetura de microsserviços com base no modelo de características. As questões estão agrupadas por características e são contextualizadas com a utilização de palavras-chave. Os níveis de atendimento às características da AMS, disponíveis na resposta de cada pergunta, são apresentados a seguir:

- **SC:** Não há conhecimento sobre o assunto;
- **SI:** Sabe-se da importância, porém o assunto não foi priorizado;
- **AISP:** Tem-se a informação, o atendimento é insatisfatório, sem plano para resolução;
- **AICP:** Tem-se a informação, o atendimento é insatisfatório, com plano para resolução;
- **AS:** Tem-se a informação e o atendimento é satisfatório; e
- **ASCP:** Tem-se a informação, o atendimento é satisfatório, com plano de mitigação.

Nr.	Característica: ESCALABILIDADE	SC	SI	AISP	AICP	AS	ASCP
1	**Demanda de recursos:** O MS consegue lidar com uma quantidade crescente de trabalho por meio da adição de recursos (*e.g.*, processamento ou memória) ao sistema?	☐	☐	☐	☐	☐	☐
2	**Crescimento qualitativo:** A escala de crescimento qualitativa (*e.g.*, quando o crescimento se dá por *page views* ou pedidos dos clientes) do MS é conhecida?	☐	☐	☐	☐	☐	☐
3	**Crescimento quantitativo:** A escala de crescimento quantitativa (*e.g.*, quantas solicitações por segundo se consegue tratar) do MS é conhecida?	☐	☐	☐	☐	☐	☐

Nr.	Característica: ESCALABILIDADE	SC	SI	AISP	AICP	AS	ASCP
4	**Aumento de dependências do MS:** Caso o MS dependa de bibliotecas, outros MS ou serviços externos, a escala de crescimento desses é conhecida?	☐	☐	☐	☐	☐	☐
5	**Aumento de dependentes do MS:** As bibliotecas, outros MS ou serviços externos de que o MS dependa, caso existam, estão preparados para um eventual crescimento?	☐	☐	☐	☐	☐	☐

Nr.	Característica: ELASTICIDADE	SC	SI	AISP	AICP	AS	ASCP
6	**Automatização:** O processo de provisionamento de recursos requerido pelo MS é automatizado?	☐	☐	☐	☐	☐	☐
7	**Recurso sob demanda:** O redimensionamento de recursos requerido pelo MS (*e.g.*, processamento ou armazenamento) é feito sob demanda?	☐	☐	☐	☐	☐	☐

Nr.	Característica: DESEMPENHO	SC	SI	AISP	AICP	AS	ASCP
8	**Requisitos de desempenho:** Os requisitos de desempenho relacionados à operação do MS foram identificados (*e.g.*, capacidade de processamento, memória ou armazenamento)?	☐	☐	☐	☐	☐	☐
9	**Requisitos de arquitetura:** Foi especificada uma proposta de atendimento aos requisitos de desempenho identificados, em nível de arquitetura?	☐	☐	☐	☐	☐	☐

Nr.	Característica: DESEMPENHO						
10	**Comportamento temporal:** O tempo de processamento de requisições e respostas (*time behavior*), bem como as taxas de transferência do MS ao executar suas funções, são conhecidos?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
11	***Hardware* exclusivo:** Existe um (segmento de) *hardware* dedicado especificamente ao MS?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: DISPONIBILIDADE						
12	**Acordo de nível de serviço:** O período de tempo em que o MS deve estar operacional e acessível foi definido?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
13	**Utilização:** O MS está operacional e acessível quando necessário (*i.e.*, dentro de um período de tempo)?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
14	**Características não-funcionais:** A disponibilidade avaliada é capaz de atender às demandas do domínio e do negócio aos quais o MS está associado?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: AUTONOMICIDADE						
15	**Isolamento:** O MS opera em um processo próprio, executando suas tarefas sem depender de outros serviços?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: DESCOBERTA DE SERVIÇO	
16	**Inicialização do MS:** A instância do MS, em sua inicialização, se registra (*e.g.*, via *host* e endereço IP) automaticamente no registro de serviços (*Service Registry*), tornando-se disponível para descoberta?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐
17	**Encerramento do MS:** A instância do MS é cancelada automaticamente no processo de encerramento no registro de serviços (*Service Registry*)?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐
18	**Monitorabilidade:** A descoberta de serviços *(Service Discovery)* está interligada com o monitoramento do MS, de modo que uma instância obtenha e mantenha um registro somente após passar com êxito em seus testes de monitoramento, *i.e.*, verificação de integridade ou verificação de saúde?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐

Nr.	Característica: IMPLANTABILIDADE	
19	**Automatização:** Existe um grau de automatização satisfatório no processo de implantação do MS?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐
20	**Replicação:** A capacidade de replicação do MS é conhecida?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐
21	**Independência:** O MS é implantado de forma independente de outros serviços?	SC SI AISP AICP AS ASCP ☐ ☐ ☐ ☐ ☐ ☐

Nr.	Característica: VIRTUALIZAÇÃO	SC	SI	AISP	AICP	AS	ASCP
22	**Tempo de inicialização:** O tempo de inicialização da máquina virtual que contém o MS é viável ao domínio e ao negócio aos quais o MS está associado?	☐	☐	☐	☐	☐	☐
23	**Ativação e desativação de MS:** As instâncias do MS são iniciadas e interrompidas da mesma maneira?	☐	☐	☐	☐	☐	☐
24	**Armazenamento**: Caso o MS esteja sendo executado em um *host* contendo múltiplas máquinas virtuais, o espaço necessário para a execução e a taxa de crescimento das instâncias são conhecidas, de forma a evitar problemas de escassez de espaço em disco?	☐	☐	☐	☐	☐	☐
25	**Conteinerização:** O MS utiliza contêineres para a virtualização?	☐	☐	☐	☐	☐	☐
26	**Análise de dependências**: As dependências (*e.g.*, sistema operacional, códigos, *softwares*, arquivos e dependências) necessárias para o MS executar estão empacotados em uma imagem da máquina virtual ou do contêiner, permitindo uma implantação padronizada?	☐	☐	☐	☐	☐	☐

Nr.	Característica: MANUTENIBILIDADE	SC	SI	AISP	AICP	AS	ASCP
27	**Analisabilidade:** É possível avaliar o impacto de uma alteração pretendida no MS ou diagnosticar deficiências ou causas de falhas?	☐	☐	☐	☐	☐	☐

Nr.	Característica: MANUTENIBILIDADE	SC	SI	AISP	AICP	AS	ASCP
28	**Modificabilidade:** O MS pode ser modificado para melhorias, correções ou adaptações a eventuais mudanças?	☐	☐	☐	☐	☐	☐
29	**Propagação de defeitos:** A estrutura do MS permite que alterações evitem a propagação de defeitos ou a degradação da qualidade do sistema?	☐	☐	☐	☐	☐	☐

Nr.	Característica: AUTOMATICIDADE	SC	SI	AISP	AICP	AS	ASCP
30	**Gerenciamento de dependências:** O MS utiliza alguma ferramenta de gerenciamento de dependências, como NPM, Pip, NuGet, Composer ou Maven?	☐	☐	☐	☐	☐	☐
31	**Versionamento de código:** Um sistema gerenciador de código-fonte está sendo utilizado para armazenar e controlar as versões do código-fonte do MS?	☐	☐	☐	☐	☐	☐
32	**Repetibilidade de processo:** O processo de entrega de funcionalidades do MS é automatizado e passível de repetição?	☐	☐	☐	☐	☐	☐
33	**Frequência de entregas:** A frequência com a qual novas funcionalidades do MS são entregues em um determinado ambiente é satisfatória?	☐	☐	☐	☐	☐	☐

Nr.	Característica: CONTINUIDADE	SC	SI	AISP	AICP	AS	ASCP
34	**Automatização:** A automatização da IC (Integração Contínua ou *Continuous Integration*) atende aos objetivos técnicos?	☐	☐	☐	☐	☐	☐
35	**Pipeline:** O MS possui um *pipeline* de IC (Integração Contínua ou *Continuous Integration*) e, caso aplicável, Entrega Contínua (*Continuous Delivery*)?	☐	☐	☐	☐	☐	☐
36	**Estratégia de integração:** A estratégia de integração de alterações do MS no sistema de controle de versão é realizada por meio de IC?	☐	☐	☐	☐	☐	☐
37	**Notificação de resultados:** A notificação dos resultados (*e.g.*, de testes automatizados ou de *builds*) aos *stakeholders* é automática?	☐	☐	☐	☐	☐	☐
38	**Entrega e Implantação Contínua:** O código do MS que passou nos testes automatizados está sempre pronto para ser entregue ou implantado em um determinado ambiente de forma rápida e segura?	☐	☐	☐	☐	☐	☐

Nr.	Característica: OBSERVABILIDADE	SC	SI	AISP	AICP	AS	ASCP
39	**Compreensibilidade do estado:** É possível compreender o estado interno do MS a partir de suas saídas?	☐	☐	☐	☐	☐	☐
40	**Representação:** É possível observar o funcionamento do MS por meio de algum mecanismo (*e.g.*, *dashboards*, gerenciadores de tráfego, notificações e *logs*)?	☐	☐	☐	☐	☐	☐

Nr.	Característica: OBSERVABILIDADE

Nr.	Característica: MONITORABILIDADE	SC	SI	AISP	AICP	AS	ASCP
41	**Verificação de saúde:** Há meios de se verificar o funcionamento do MS, por algum comando de checagem da saúde da API (*API health*)?	☐	☐	☐	☐	☐	☐
42	**Efetividade:** O monitoramento do MS quanto a erros, a alertas e ao desempenho garante a sua efetividade?	☐	☐	☐	☐	☐	☐
43	**Notificação:** Há alertas para as principais métricas do MS?	☐	☐	☐	☐	☐	☐
44	**Acionabilidade:** Os alertas referentes ao MS são acionáveis?	☐	☐	☐	☐	☐	☐
45	**Personalização de alertas:** É possível configurar alertas para serem acionados quando determinadas mensagens aparecerem nos registros de *log* do MS?	☐	☐	☐	☐	☐	☐
46	***Logs*:**É possível pesquisar e analisar registros de *log* referentes ao MS?	☐	☐	☐	☐	☐	☐

Nr.	Característica: TESTABILIDADE	SC	SI	AISP	AICP	AS	ASCP
47	**Critérios de teste:** Os critérios de teste podem ser estabelecidos com eficácia para o MS?	☐	☐	☐	☐	☐	☐

Nr.	Característica: TESTABILIDADE	SC	SI	AISP	AICP	AS	ASCP
48	**Eficácia:** A realização de testes do MS atende aos critérios de testes com eficácia?	☐	☐	☐	☐	☐	☐
49	**Identificação de falhas:** Os testes associados ao MS conseguem identificar falhas?	☐	☐	☐	☐	☐	☐
50	**Simulação de ambiente:** O MS utiliza um ambiente de testes que replica suficientemente as nuances do ambiente de produção?	☐	☐	☐	☐	☐	☐
51	**Testes contínuos:** Os testes de unidade do MS são realizados a cada modificação?	☐	☐	☐	☐	☐	☐

Nr.	Característica: CONFIABILIDADE	SC	SI	AISP	AICP	AS	ASCP
52	**Credibilidade:** O MS desempenha suas funções nas condições determinadas sem falhas?	☐	☐	☐	☐	☐	☐
53	**Desempenho:** O MS desempenha suas funções atendendo aos requisitos de desempenho?	☐	☐	☐	☐	☐	☐
54	**Roteamento:** Quando uma instância de um MS em execução não consegue lidar com uma requisição, ocorre o roteamento para outras instâncias em funcionamento?	☐	☐	☐	☐	☐	☐

Nr.	Característica: RESILIÊNCIA	SC	SI	AISP	AICP	AS	ASCP
55	**Tolerância a falhas:** O MS opera conforme pretendido, mesmo na presença de falhas de *hardware* ou de *software*?	☐	☐	☐	☐	☐	☐
56	**Propagação:** Falhas associadas ao MS não afetam o restante da aplicação?	☐	☐	☐	☐	☐	☐
57	**Prevenção de falhas:** O MS utiliza mecanismos (*i.e.*, *Circuit Breaker*) para impedir que uma falha de rede ou serviço se espalhe para outros serviços, impedindo que outros MS façam solicitações desnecessárias ou inadequadas?	☐	☐	☐	☐	☐	☐
58	**Recuperabilidade:** No caso de uma interrupção ou falha, o MS pode recuperar os dados diretamente afetados e restabelecer o estado dos sistemas que estavam utilizando-o antes da ocorrência da falha?	☐	☐	☐	☐	☐	☐

Nr.	Característica: SEGURANÇA	SC	SI	AISP	AICP	AS	ASCP
59	**Confidencialidade:** O MS garante que os dados sejam acessíveis apenas àqueles autorizados a ter acesso?	☐	☐	☐	☐	☐	☐
60	**Integridade:** O MS impede o acesso não autorizado ou a modificação dos dados sensíveis?	☐	☐	☐	☐	☐	☐
61	**Não Repúdio:** É possível provar que as ações ou os eventos do MS aconteceram, de modo que eles não possam ser repudiados posteriormente?	☐	☐	☐	☐	☐	☐

Nr.	Característica: SEGURANÇA						
62	**Rastreabilidade:** As ações do MS podem ser rastreadas exclusivamente?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
63	**Autenticidade:** A identidade do MS pode ser comprovada?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: COMUNICABILIDADE						
64	**Comunicação eficiente:** O MS é capaz de se comunicar com outros MS de forma consistente, sem perda de confiança no processo?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
65	**Interface:** A comunicação do MS é feita por meio de contratos bem definidos, sem a exposição de detalhes de implementação?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
66	**Compatibilidade:** O MS pode trocar informações com outros MS, componentes ou sistemas e usar as informações trocadas?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐
67	**Estratégia de comunicação:** No caso de padrão de comunicação assíncrono, o MS faz uso de filas para armazenar as requisições para serem tratadas no momento devido?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: COREOGRAFIA						
68	**Descentralização:** O MS sabe exatamente quando executar suas operações e com que serviços interagir, sem que haja um controle centralizado?	SC ☐	SI ☐	AISP ☐	AICP ☐	AS ☐	ASCP ☐

Nr.	Característica: COREOGRAFIA	SC	SI	AISP	AICP	AS	ASCP
69	**Delegação:** A capacidade de tomada de decisão do MS em relação ao fluxo de trabalho é distribuída entre os MS com os quais ele interage?	☐	☐	☐	☐	☐	☐

Nr.	Característica: ORQUESTRAÇÃO	SC	SI	AISP	AICP	AS	ASCP
70	**Coordenação:** O MS apresenta fluxos de trabalhos complexos com outros MS, exigindo níveis de coordenação e de sincronização?	☐	☐	☐	☐	☐	☐
71	**Responsabilidade:** O MS se preocupa única e exclusivamente da tarefa pela qual é responsável, não ocupando-se de outros aspectos do fluxo geral de trabalho?	☐	☐	☐	☐	☐	☐

Nr.	Característica: MODULARIDADE	SC	SI	AISP	AICP	AS	ASCP
72	**Baixo acoplamento:** O MS é composto por componentes discretos (coesos e fracamente acoplados), de modo que mudanças em um componente tenham impacto mínimo sobre outros?	☐	☐	☐	☐	☐	☐
73	**Componibilidade:** O MS pode ser separado e recombinado com o objetivo de expandir funcionalidades?	☐	☐	☐	☐	☐	☐

Nr.	Característica: REUSABILIDADE	SC	SI	AISP	AICP	AS	ASCP
74	**Desenvolvimento para reutilização:** O MS pode ser reaproveitado em mais de um sistema ou na construção de outros MS?	☐	☐	☐	☐	☐	☐
75	**Desenvolvimento com reutilização:** O MS faz o reúso de outros sistemas ou MS em sua construção?	☐	☐	☐	☐	☐	☐

Nr.	Característica: COMPONIBILIDADE	SC	SI	AISP	AICP	AS	ASCP
76	**Flexibilidade:** A funcionalidade do MS pode ser consumida de diferentes maneiras e para diferentes propósitos? Ou seja, o MS pode ser recombinado para o atendimento a objetivos diversos?	☐	☐	☐	☐	☐	☐

Nr.	Característica: ESPECIFICIDADE	SC	SI	AISP	AICP	AS	ASCP
77	**Responsabilidade única:** O MS possui uma responsabilidade única, atômica e bem definida?	☐	☐	☐	☐	☐	☐

Nr.	Característica: COESÃO	SC	SI	AISP	AICP	AS	ASCP
78	**Granularidade:** As operações fornecidas pelo MS atendem a apenas uma funcionalidade?	☐	☐	☐	☐	☐	☐
79	**Isolamento:** A alteração de comportamento do MS, quando necessária, é realizada em somente um local?	☐	☐	☐	☐	☐	☐

Nr.	Característica: COESÃO

Nr.	Característica: ACOPLAMENTO						
		SC	SI	AISP	AICP	AS	ASCP
80	**Baixa dependência:** As dependências e conexões do MS em relação a outros limitam-se às estritamente necessárias?	☐	☐	☐	☐	☐	☐
81	**Encapsulamento:** A mudança no MS ocorre de forma a exigir o mínimo possível de alterações em outros MS?	☐	☐	☐	☐	☐	☐

Nr.	Característica: TAMANHO						
		SC	SI	AISP	AICP	AS	ASCP
82	**Granularidade fina:** As funcionalidades do MS são atômicas, *i.e.*, não podem ser decompostas em outras de menor tamanho?	☐	☐	☐	☐	☐	☐

Nr.	Característica: PROPRIEDADE DOS DADOS						
		SC	SI	AISP	AICP	AS	ASCP
83	**Recursos próprios:** O MS utiliza um banco de dados exclusivo aos seus propósitos?	☐	☐	☐	☐	☐	☐

Nr.	Característica: DISTRIBUTIVIDADE	SC	SI	AISP	AICP	AS	ASCP
84	**Transparência:** O MS pode ser considerado "invisível" (*i.e.*, em termos de acesso, de falha, de localização, de realocação, de replicação, de persistência e de concorrência) aos sistemas usuários?	☐	☐	☐	☐	☐	☐

Nr.	Característica: BALANCEAMENTO DE CARGA	SC	SI	AISP	AICP	AS	ASCP
85	**Roteamento de requisições:** Na utilização do MS, o tráfego de entrada está sendo distribuído por meio de um balanceador de carga, com a adição ou a remoção de máquinas, dependendo do tráfego e das falhas?	☐	☐	☐	☐	☐	☐
86	**Gerenciamento de tráfego:** Existe uma estratégia para analisar o padrão de tráfego referente ao MS?	☐	☐	☐	☐	☐	☐
87	**Automatização:** Na utilização do MS, o equilíbrio da capacidade de recursos necessária na adição de um novo servidor ao *cluster* é realizada de forma automática?	☐	☐	☐	☐	☐	☐

APÊNDICE D – Dados da avaliação do modelo de características

Este apêndice apresenta os dados coletados na avaliação do modelo de características e na avaliação do glossário de termos.

Os participantes registraram observações gerais e específicas avaliando o modelo de características por meio do *checklist* FMCheck.

Observações gerais dos participantes

Observação do participante 3: "Em geral o modelo tem boas caracteristicas mas ainda precisa de alguns ajustes e melhoramentos. Algumas características de microserviços estão sobrepostas às características do ambiente de operação."

Respostas e observações específicas dos participantes

A Tabela 41 a seguir apresenta as respostas e observações dos participantes quanto aos itens de verificação individual das características do modelo. As observações estão dispostas após cada pergunta.

Tabela 41 - Respostas e observações: itens de verificação individual das características do modelo.

#	**Item de verificação**	**P1**	**P2**	**P3**	**P4**
1	Todas as características do modelo foram descritas com clareza e estão corretas?	Sim	Sim	Não	Sim
Observação do participante 3: "Algumas caracteristicas descritas no modelo não fazem parte do microserviço em si, mas a infrastrutura por trás de um ambiente orchestrado para microserviços. O modelo passa uma impressão de que o microserviço conhece essas caracteristicas que são impostas em muitas vezes pelo orquestrador, mas isso não é verdade." Observação do participante 4: "Não sei se distributividade é uma característica, pois é inerente à MSA."					
2	A opcionalidade/obrigatoriedade das características do modelo estão em conformidade com o descrito pelo domínio?	Sim	Sim	Não	Sim

continuação na próxima página

Observação do participante 3: “Acredito que a maioria das características são desejáveis ao contrário de obrigatoriedade.” Observação do participante 4: “Parece que sim, mas não tem explicação de como foi definido qual característica é obrigatório, opcional, ou alternativa. Por exemplo, não sei se descoberta de serviços é obrigatório.”					
3	É possível identificar o tipo de cada característica do modelo a partir de sua descrição?	Sim	Sim	Sim	Sim
Observação do participante 4: “Na sua maioria, eu acredito que sim.”					
4	As características que representam conceitos reais do domínio estão devidamente representadas no modelo como *características de domínio conceituais (conceptual)*?	Sim	Sim	Nsa	Sim
5	As características que representam funcionalidades do domínio estão devidamente representadas no modelo como *características de domínio funcionais (functional)*?	Sim	Sim	Nsa	Sim
6	As características que representam uma entidade real do domínio estão devidamente representadas no modelo como *características de entidade*?	Sim	Sim	Nsa	Sim
7	As características que representam atributos de um ambiente relacionado ao uso/operação da aplicação do domínio estão devidamente representadas no modelo como *características do ambiente operacional*?	Sim	Sim	Nsa	Sim
8	As características que representam uma tecnologia utilizada para modelar ou implementar o domínio estão devidamente representadas no modelo como *características de tecnologia do domínio*?	Sim	Sim	Nsa	Sim
9	As características que representam uma tecnologia utilizada para implementar outras características do modelo estão devidamente representadas no modelo como *características da técnica de implementação*?	Sim	Sim	Nsa	Nsa
Observação do participante 4: “Respondi, Nsa, pois não sei responder essa pergunta.”					
10	As características que não possuem ligações concretas com o domínio, mas facilitam o seu entendimento, estão representadas no modelo como *características organizacionais*?	Sim	Sim	Nsa	Não

continuação na próxima página

11	Alguma característica do modelo, embora correta, está fora do escopo do modelo, não contribuindo para o entendimento do domínio?	Não	Não	Nsa	Sim
Observação do participante 4: "Não sei se distributividade é uma característica, pois é inerente à MSA."					
12	Existem características distintas no modelo que representam um mesmo conceito do domínio?	Não	Não	Nsa	Não
13	Algum conceito relevante do domínio deixou de ser incluído no modelo?	Não	Não	Não	Não

Fonte: O autor, 2023.

A seguir, a Tabela 42 apresenta as respostas e observações dos participantes quanto aos itens de verificação dos relacionamentos entre as características do modelo. As observações estão dispostas após cada pergunta.

Tabela 42 - Respostas e observações: itens de verificação dos relacionamentos entre as características do modelo.

#	**Item de verificação**	**P1**	**P2**	**P3**	**P4**
14	As situações do domínio em que uma ou mais de uma característica podem ser escolhidas dentro de um conjunto de características (OU, OU Exclusivo) estão devidamente representadas no modelo?	Sim	Sim	Nsa	Sim
Observação do participante 4: "Eu imagino que sim, mas eu não entendi muito bem a existência do relacionamento obrigatório/opcional dentro de alternativo. Isso ficou confuso."					
15	As cardinalidades dos pontos de variação do modelo estão corretas?	Sim	Sim	Nsa	Nsa
16	Os pontos de variação do modelo estão descritos com clareza e suas descrições refletem as características que os compõem?	Sim	Sim	Nsa	Sim
17	Duas ou mais características do modelo estão reunidas em um relacionamento, mas não é possível identificar este relacionamento no domínio?	Não	Não	Nsa	Não
18	Existe algum relacionamento entre características que deixou de ser informado no modelo?	Sim	Não	Nsa	Não
Observação P1: "A relação entre modularidade e manutenibilidade."					

continuação na próxima página

19	A hierarquia entre as características está em conformidade com o domínio?	Sim	Não	Não	Sim
Observação do participante 2: “Segundo o glossário, a característica de disponibilidade está relacionada diretamente a escalabilidade, mas na descrição de disponibilidade é dito que ‘De acordo com Becker(2019), a alta disponibilidade de uma AMS está relacionada à capacidade de replicação dos microsserviços favorecendo a elasticidade’, o que me leva a crer que a disponibilidade deveria estar relacionada com elasticidade e não diretamente com escalabilidade.” Observação do participante 3: “A elasticidade no mesmo nível de desempenho e disponibilidade não faz muito sentido. Eu vejo a elasticidade como um modo de obter/manter o desempenho e disponibilidade utilizando recursos necessários.” Observação do participante 4: “Fiquei na dúvida se escalabilidade e desempenho não deveriam ser trocadas de lugar uma com a outra.”				
20	Alguma característica foi indevidamente apontada como generalização de outra?	Não	Não	Nsa	Não
21	As características que estão apontadas no modelo como “implementada por” outra característica, possuem este relacionamento no domínio?	Sim	Sim	Nsa	Sim
22	Os relacionamentos de agregação e de composição entre características do domínio apontados no modelo condizem com a realidade deste domínio?	Sim	Sim	Nsa	Nsa
Observação do participante 4: “Não sei qual é a relação de agregação e de composição no FM.”					
23	Alguma dependência ou relação de mútua exclusividade entre características no modelo não se aplica ao domínio descrito?	Não	Não	Nsa	Nsa
Observação do participante 4: “O relacionamento no FM está um pouco estranho. A princípio a semântica do relacionamento é do grupo "OR", mas a legenda é do "XOR".”					
24	Alguma dependência ou relação de mútua exclusividade entre características deixou de ser informada no modelo?	Não	Não	Nsa	Não
25	A presença de alguma característica no modelo contraria outra característica do mesmo modelo?	Não	Não	Nsa	Não
26	A característica-raiz auxilia a compreensão do domínio que ela e as demais características do modelo buscam descrever?	Sim	Sim	Nsa	Sim

continuação na próxima página

27	De um modo geral, é possível compreender o domínio a partir de suas características?	Sim	Sim	Sim	Sim
28	O modelo descreve o domínio em um nível de detalhamento adequado para ser compreendido sob a perspectiva pretendida?	Sim	Sim	Não	Sim
29	O modelo apresenta as características necessárias e suficientes para orientar sua implementação?	Sim	Sim	Sim	Não
Observação do participante 4: “Apresenta uma visão geral, mas não ao nível de detalhes para a implementação.”					

Fonte: O autor, 2023.

APÊNDICE E – Dados da avaliação do glossário de termos

Este apêndice apresenta os dados coletados na avaliação do glossário de termos.

Os participantes registraram observações gerais e específicas avaliando o glossário de termos em relação à pertinência de cada item para explicar as características do modelo de características.

Observações gerais dos participantes

Observação do participante 2: "O texto de conceituação das características é muito denso e cansativo de ser lido. Há um desbalanceamento geral das definições. Algumas muito sucintas e outras muito extensas. Vi que todos os relacionamentos explorados foram via de mão única de baixo para cima. Mas algumas características contribuem tb para outras no sentido horizontal. Ou seja, existem tb alguns relacionamentos entre características não necessariamente somente hierárquico. Vocês viram isso?"

Observações específicas dos participantes

As observações de cada participante em relação às características do glossário de termos são apresentadas a seguir.

Quanto à Elasticidade

Observação do participante 1: "Acredito que vale citar que a elasticidade vai além da capacidade de replicação dos microsserviços, pois isso vale apenas para a estratégia de escalabilidade horizontal. No caso da vertical, se faz necessário a capacidade de alterar o poder computacional do *hardware* onde o microsserviço está sendo executado."

Quanto ao Desempenho

Observação do participante 1: "Acredito que utilizar REST como exemplo utilizado para mecanismos leves de comunicação está ligeiramente equivocado. Esse padrão foi criado com o foco de garantir a interoperabilidade entre sistemas e facilitar a capacidade

de alterar o estado de seus recursos/dados. Talvez utilizar como exemplo mecanismos mais focados em otimizar a comunicação entre microservicos seja interessante (*e.g.*, gRPC)."

Quanto à Disponibilidade

Observação do participante 1: " 'De acordo com Becker (2019), a alta disponibilidade de uma AMS está relacionada à capacidade de replicação dos microsserviços favorecendo a elasticidade.'

Vale ressaltar que a capacidade de replicação apenas não garante a elasticidade. Outros aspectos como dependência de sistemas externos e mecanismos de resiliência (*e.g.*, *circuit breaker*) também tem um papel importante em garantir a disponibilidade."

Observação do participante 2: "Há evidência de que este conceito remete a elasticidade: 'De acordo com Becker (2019), a alta disponibilidade de uma AMS está relacionada à capacidade de replicação dos microsserviços favorecendo a elasticidade.'."

Observação do participante 3: " 'Both performance and availability can be increased if the number of servers is increased. However, this architectural decision can negatively impact the security of the system, because increasing number of servers, increases potential points of attack and failures (ROY; GRAHAM, 2008)'

Discordo da informação sobre aumento de pontos de attack e falha. Para mim a replicação de um microserviços não altera esses compenentes, uma vez que eles possuem exatamente o mesmo code base."

Quanto à Descoberta de serviço

Observação do participante 1: "Vale ressaltar que a utilização de mecanismos de *service discovery* não é necessária, mas - sim - muito útil e uma boa prática. Uma estratégia alternativa é a utilização de DNS interno à rede de microsservicos, tal mecanismo é oferecido '*out of the box*' pelo Kubernetes (https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/). Alguns autores caracterizam essa abordagem também como *service discovery* e outros não, mas acho que vale ressaltar essa alternativa no texto."

Observação do participante 3: "Achei que a definição ficou abaixo em relação a exemplos de como isso é facilmente coberto com orquestradores de microserviços hoje em dia."

Quanto à Continuidade

Observação do participante 1: " 'A continuidade, refere-se à utilização das práticas de integração contínua (IC) e de entrega contínua (EC)', a continuidade vai além dessas duas categorias. Existe um tópico que vem ganhando muita atenção nos últimos anos chamado de *Continuous Software Engineering* que investiga as diversas técnicas e métodos da continuidade no desenvolvimento de *software*. Indico o artigo *Continuous software engineering and beyond: trends and challenges*, Brian Fitzgerald and Klaas-Jan Stol, 2014."

Observação do participante 2: "Não deveria estar relacionada com Implantabilidade?"

Observação do participante 3: "O termo continuidade no português fica meio estranho para explicar praticas de devops, mas a explicação está ok."

Quanto à Observabilidade

Observação do participante 3: "Essa é uma das principais características de um microservice e tem que ser melhor explorada. Ela vai ser utilizada vários outros locais."

Quanto à Testabilidade

Observação do participante 3: "Senti falta de exemplos de tipos de testes."

Quanto à Resiliência

Observação do participante 1: "Acho que vale acrescentar que também existem padrões de *design*/arquitetura que aumentam a resiliência de um microsservico (*e.g.*, *circuit breaker*, *timeout pattern*, *retry pattern*)."

Quanto à Segurança

Observação do participante 3: "Acho que esse tópico merece uma atenção especial. Segurança em microsserviços possui diferentes características."

Quanto à Orquestração

Observação do participante 3: “É preciso trabalhar mais nesse tópico, um dos principais motivos da adoção de microsserviços é o serviço que os orquestradores executam.”

Quanto ao Tamanho

Observação do participante 1: “Acho que vale explicar um pouco melhor o porquê da seguinte afirmação ‘o pequeno tamanho traz grandes benefícios em termos de manutenibilidade e extensibilidade’.”

Observação P3: “Não ficou claro da importância dessa característica.”

Quanto à Propriedade dos dados

Observação do participante 1: “A frase ‘é possível que alguns serviços não utilizem um banco de dados’ pareceu-me um pouco confusa. Não entendi muito bem se quer dizer que não utilizam banco de dados nenhum ou que não utilizam banco de dados compartilhado entre si.”

Quanto ao Balanceamento de carga

Observação do participante 3: “Essa característica faz parte do orquestrador.”

APÊNDICE F – Dados da avaliação do questionário

Este apêndice apresenta os dados coletados na avaliação do questionário de avaliação.

Os participantes registraram observações gerais e específicas na avaliação deste artefato.

Observações gerais dos participantes

Observação do participante 7: "O questionário é muito extenso. Com alguma flexibilização (do número de características avaliadas) ele poderia ser usado como um DoR (*Definition of Ready*) para a arquitetura. Sempre que a área de negócio necessitar desenvolver um produto novo e esse produto necessitar de um ecossistema de microsserviços, cada microsserviço no desenho da solução seria avaliado pelo questionário, fazendo parte do processo de desenvolvimento, da cultura da Squad."

Observação do participante 10: "Questionário muito longo."

Observação do participante 15: O participante destacou que na sessão de virtualização não há perguntas sobre o tipo de hospedagem do MS (*e.g.*, nuvem interna ou publica). Outro ponto observado se refere à existência de perguntas sobre IaaS (*Infrastructure as a Service*) e CaaS (*Container-as-a-Service*), mas não sobre SaaS (*Software as a Service*) e PaaS (*Platform-as-a-Service*).

Observações específicas dos participantes

As observações de cada participante em relação às perguntas do questionário de avaliação são apresentadas a seguir.

Quanto à Escalabilidade

1) **Demanda de recursos**: O MS consegue lidar com uma quantidade crescente de trabalho por meio da adição de recursos (*e.g.*, processamento ou memória) ao sistema?

Observação do participante 9: “Para avaliar a escalabilidade, pode ser interessante não só mencionar a adição de recursos quando necessário (criação de mais réplicas dos microsserviços), como também a remoção de recursos quando não necessário (remoção de réplicas sem uso).”

Quanto à Manutenibilidade

28) **Modificabilidade**: O MS pode ser modificado para melhorias, correções ou adaptações a eventuais mudanças?

Observação do participante 15: “Não tenho certeza se essa pergunta é pertinente. Qualquer aplicação deveria ser aberta a correções e mudanças. Mas tem o caso de aplicações que foram compradas como pacote fechado, talvez nesses casos possa ser difícil alterar ou evoluir.”

Quanto ao Acoplamento

81) **Encapsulamento**: A mudança no MS ocorre de forma a exigir o mínimo possível de alterações em outros MS?

Observação do participante 11: “Me parece que o conceito de encapsulamento não é o mais adequado para essa pergunta. A pergunta feita tem mais relação com o conceito de baixo acoplamento.”

APÊNDICE G – Dados da avaliação da técnica de visualização

Este apêndice apresenta os dados coletados na avaliação da técnica de visualização.

Os participantes registraram suas observações de forma verbal por meio do protocolo *think-aloud*. Esses dados foram transcritos para o formato de texto por meio da ferramenta *Transkriptor* (https://app.transkriptor.com/). Após análise dos trechos, foram identificadas as unidades de significado, permitindo a categorização das informações.

Observações dos participantes

Observação do participante 16: A técnica de visualização tem métricas interessantes, mas eu acho que possui detalhes demais e na prática não vemos tudo isso e também isso teria que ser aplicado pra cada microsserviço e você tem microsserviços que você criou o código-fonte então você trata de uma forma, mas há microsserviços que acessam serviços de terceiros, que são tratados de outra maneira, então não sei se isso se aplica a todo o caso.

Duração do áudio do participante 16: trinta e oito segundos.

Pontos-chave identificados com a observação do participante 16:

1. Relevância das métricas:
    - O participante expressa que a técnica de visualização possui métricas interessantes.
2. Detalhamento excessivo:
    - O participante destaca que técnica de visualização possui detalhes demais, os quais na prática não são utilizados.
3. Incerteza sobre a aplicabilidade geral:
    - O participante questiona a aplicabilidade em um contexto geral, pois há microsserviços que acessam serviços de terceiros.

Observação do participante 17: A visualização do resultado da avaliação é bastante satisfatória, pois apresenta os atributos do microsserviço por meio de analogias com medidores que permitem uma mensuração bastante rápida. A visão de cada atributo é bem concisa e permite saber se foi atingido um estado satisfatório ou ideal para aquele atributo ou se falta um alcance de um estado mínimo. Então fiquei bastante satisfeito

com a forma de visualização e recomendaria fortemente a adoção desse tipo de visualização nos processos de acompanhamento do desenvolvimento de microsserviços tanto no processo de construção de novos projetos quanto na manutenção daqueles existentes.

Duração do áudio do participante 17: um minuto e vinte e sete segundos.

Pontos-chave identificados com a observação do participante 17:

1. Satisfação com a visualização do resultado da avaliação:
    - O participante expressa satisfação com a forma como os resultados da avaliação são visualizados.
    - A visualização é descrita como bastante satisfatória, indicando que o participante aprovou a apresentação das características de microsserviços por meio de analogias a medidores (gauges).
2. Rapidez e clareza na mensuração dos atributos:
    - O participante destaca que a visualização permite uma mensuração rápida e concisa em relação às características da arquitetura de microsserviços.
    - A visão de cada característica é descrita como concisa, o que possibilita uma avaliação clara e rápida do estado da característica.
3. Identificação de estados satisfatórios e mínimos:
    - O participante menciona que a visualização permite saber se um atributo atingiu um estado satisfatório ou ideal.
    - Também é destacado que a visualização permite identificar se há falta de alcance de um estado mínimo para determinada característica.
4. Recomendação para adoção da visualização:
    - O participante expressa recomendação para a adoção desse tipo de visualização nos processos de acompanhamento do desenvolvimento de microsserviços, tanto no processo de construção de novos projetos, quanto na manutenção de projetos existentes.

Observação do participante 18: Com relação a a ferramenta, temos um diagrama, um mapa muito legível, que está bem relacionado com os requisitos que um microsserviço precisa pra atender conceitualmente a definição de microsserviço. Está bem bacana, coeso, legível e bem estruturado. A questão de utilizar uma árvore, você tem uma hierarquia entre os conceitos e um ponto que eu não não consegui entender muito bem é

o que os gauges representam, a escalabilidade tem 60 enquanto cada filho tem 100. A escalabilidade não deveria estar também em 100? Isso para mim não ficou muito claro. É fácil de usar, eu acho que dá pra identificar o que o microsserviço possa apresentar de deficiência. Eu acho que a partir dos gauges, pode-se criar planos pra ajustar aquilo que não está funcionando bem no microsserviço. A ferramenta tá aprovada na minha opinião.

Duração do áudio do participante 18: três minutos e quatro segundos.

Pontos-chave identificados com a observação do participante 18:

1. Avaliação da ferramenta:
    - O participante destaca aspectos como a legibilidade, a relação com os requisitos conceituais dos microsserviços, a estrutura bem organizada e a facilidade de uso da ferramenta.
2. Dúvidas sobre significado dos gauges:
    - O participante relatou dúvidas sobre a interpretação da avaliação dos gauges envolvendo características pais e filhas.
3. Possibilidade de identificação de deficiências:
    - O participante destaca a possibilidade de identificar deficiências em microsserviços.
4. Utilização dos gauges para criar planos de melhoria:
    - O participante sugere a utilização dos gauges como base para a criação de planos de melhorias em microsserviços.
5. Opinião positiva sobre a ferramenta:
    - O participante expressa satisfação geral com a ferramenta.

Observação do participante 19: Gostei da técnica de visualização. Acho que apresenta de forma clara os pontos que precisam ser melhorados, os pontos que estão atendendo bem ao microsserviço e acho que possui potencial pra poder apresentar mais informações.

Duração do áudio do participante 19: vinte e três segundos.

Pontos-chave identificados com a observação do participante 19:

1. Apreciação da técnica de visualização:

- O participante expressa satisfação com a técnica de visualização.
- O participante ressalta que a técnica apresenta de forma clara os pontos que precisam ser melhorados e os pontos que estão atendendo bem ao microsserviço.

2. Potencial de expansão da técnica de visualização:

   - O participante destaca que a técnica apresenta potencial para fornecer mais informações.

Observação do participante 20: Sobre a técnica de visualização a única coisa que me deixou meio confuso aqui é que parece que é uma estrutura em árvore que apresenta itens e subitens, mas as perguntas estão categorizadas, no item ou no subitem, mas o subitem não conta ou não tem peso na contagem dos itens pais. Parece que não existe peso ou relação entre pai e filho nessa hierarquia. Na minha avaliação, a Comunicabilidade possui Coreografia e Orquestração como filhas, que apresentam 0, mas a Comunicabilidade está 40. Eu entendi que a avaliação do gauge depende das respostas das perguntas, mas olhando assim entendo que se os filhos estão mal avaliados, isso poderia pesar negativamente para os pais. Talvez o ideal fosse que as características com filhos só tivessem perguntas nas folhas, e o gauge dessas características pais seria uma média ou mediana dos gauges dos filhos. Pensaria em uma organização um pouco melhor, para não ter essa confusão.

Duração do áudio do participante 20: dois minutos e vinte e sete segundos.

Pontos-chave identificados com a observação do participante 20:

1. Discrepância na avaliação entre características pais e filhas:

   - O participante destacou que os itens e subitens não parecem ter uma relação de peso ou influência entre si.

2. Sugestões de melhoria na avaliação entre características pais e filhas:

   - O participante sugere que características com filhos tenham apenas perguntas nas folhas e que a pontuação dos pais seja calculada como a média ou mediana das pontuações dos filhos.
   - O participante sugere que a avaliação dos itens pais deveria levar em consideração a avaliação dos subitens, e que a média ou mediana dos gauges dos subitens poderia ser utilizada para representar a pontuação dos itens pais.

Observação do participante 21: Eu gostei da avaliação, achei bem bem interessante. Eu acho que, dependendo do modelo do microsserviço que você está avaliando,

algumas perguntas não são aplicáveis, então talvez poderia ter uma opção não aplicável, pois acho que a opção sem conhecimento não reflete bem todos os possíveis cenários. Gostei do resultado da avaliação com esses gráficos. Esses gauges são bem interessantes, eles ajudam a identificar as características que podem ser melhoradas no microsservice. Na minha avaliação, o desempenho está em 40, o que significa que talvez eu poderia olhar melhor para o nível de desempenho. Algumas coisas são bem interessantes e refletem bem o que eu penso sobre construção de microsserviços, sobre quando exatamente usar um microsserviço. É isso aí. Achei legal, achei bem bem interessante, acho que a criação de ferramentas com base nessa avaliação ajudaria principalmente profissionais no processo de arquitetura no nível de *Solution architecture* e de *Enterprise architecture*. Bem interessante, achei legal, muito bom, gostei.

Duração do áudio do participante 21: um minuto e trinta e cinco segundos.

Pontos-chave identificados com a observação do participante 21:

1. Reação positiva à avaliação:
    - O participante expressa uma reação positiva em relação à avaliação realizada, descrevendo-a como interessante, destacando a utilidade dos gauges utilizados.
2. Inclusão de opção de resposta na avaliação:
    - O participante ressalta que algumas perguntas não são aplicáveis ao cenário sendo avaliado e sugere a inclusão da opção "não aplicável" como possibilidade de resposta.
3. Identificação de oportunidades de melhorias:
    - É destacado que os gauges ajudam a identificar as características que podem ser melhoradas em um microsserviço.
4. Valor da avaliação para arquitetos e profissionais de TI:
    - O participante reconhece o valor da avaliação não apenas para si, mas também para profissionais envolvidos em arquitetura de soluções e arquitetura empresarial.

ANEXO – FMCheck

Este anexo apresenta a técnica de inspeção baseada em *checklist*, proposta por Mello et al. (2014), conhecida como FMCheck, que visa apoiar a detecção de defeitos em *feature models*. A técnica foi desenvolvida com foco em processos de inspeção que preveem a execução de inspeções individuais, em que cada inspetor detecta e relata individualmente as discrepâncias por ele identificadas.

O inspetor não precisa ter conhecimento prévio do domínio para aplicar o FMCheck, uma vez que alguma descrição textual válida, como uma especificação de requisitos ou especificação de domínio, é definida como pré-requisito (documento base) a ser usada como base de comparação durante a aplicação da técnica.

Atividades

A aplicação do FMCheck consiste em três atividades:

1. Primeira atividade: consiste no preenchimento de um questionário de caracterização do modelo por um analista de domínio. Este questionário tem como objetivo fornecer a base para a configuração da lista de verificação, eliminando itens de avaliação do FMCheck que não precisam ser aplicados em um cenário de verificação específico, evitando esforços desnecessários, já que o FMCheck fornece itens que podem ser aplicados em contextos mais amplos, abrangendo várias notações. É importante destacar que o questionário deve ser preenchido com base no que é fornecido pela notação do modelo de características adotado e não apenas com base nas características apresentadas no modelo a ser inspecionado;

2. Segunda atividade: consiste na configuração da lista de verificação, a ser feita pelo moderador da inspeção, com o auxílio de uma tabela de rastreabilidade que relaciona cada resposta do questionário de caracterização do modelo a itens específicos de avaliação da lista de verificação; e

3. Terceira atividade: consiste em realizar uma ou mais inspeções individuais, produzindo um ou mais relatórios de discrepância que descrevem cada discrepância, sua categoria de defeito e localização.

O *Checklist*

O *checklist* da referida técnica é composto por trinta e quatro itens de verificação. Para cada item de verificação, existem três respostas possíveis: “Sim”, “Não” ou “N.A.” (não aplicável). Uma inspeção específica poderá conter desde um número mínimo de itens de verificação (14), aplicável a qualquer *feature model*, até todos os 34 itens de verificação, que estão divididos em três grupos de verificação:

1. *Verificação individual das características do modelo*: São treze itens de verificação que tratam exclusivamente da análise de cada característica do modelo, sem observar os seus relacionamentos, buscando garantir que o modelo possui características corretas, pertinentes ao domínio, suficientemente abrangentes e descritas com clareza e objetividade.

2. *Verificação dos relacionamentos entre as características do modelo*: São dezesseis itens de verificação que orientam o inspetor a examinar os relacionamentos entre as características, analisando o quanto os relacionamentos entre as características do modelo o tornam compreensível, aderente ao domínio e implementável; e

3. *Verificação das regras de composição do modelo*: São cinco itens de verificação que orientam o inspetor no exame da clareza, completude, corretude, pertinência e consistência das regras de composição do modelo com relação aos conceitos do domínio.

A seguir, os treze itens do grupo “Verificação individual das características do modelo” são apresentados na Tabela 43.

Tabela 43 - Itens para verificação individual das características do modelo.

#	**Item de verificação**	**Resposta**
1	Todas as características do modelo foram descritas com clareza e estão corretas?	Sim ( ) Não ( ) N.A. ( )
2	A opcionalidade/obrigatoriedade das características do modelo estão em conformidade com o descrito pelo domínio?	Sim ( ) Não ( ) N.A. ( )
3	É possível identificar o tipo de cada característica do modelo a partir de sua descrição?	Sim ( ) Não ( ) N.A. ( )

continuação na próxima página

4	As características que representam conceitos reais do domínio estão devidamente representadas no modelo como *características de domínio conceituais (conceptual)*?	Sim ( ) Não ( ) N.A. ( )
5	As características que representam funcionalidades do domínio estão devidamente representadas no modelo como *características de domínio funcionais (functional)*?	Sim ( ) Não ( ) N.A. ( )
6	As características que representam uma entidade real do domínio estão devidamente representadas no modelo como *características de entidade*?	Sim ( ) Não ( ) N.A. ( )
7	As características que representam atributos de um ambiente relacionado ao uso/operação da aplicação do domínio estão devidamente representadas no modelo como *características do ambiente operacional*?	Sim ( ) Não ( ) N.A. ( )
8	As características que representam uma tecnologia utilizada para modelar ou implementar o domínio estão devidamente representadas no modelo como *características de tecnologia do domínio*?	Sim ( ) Não ( ) N.A. ( )
9	As características que representam uma tecnologia utilizada para implementar outras características do modelo estão devidamente representadas no modelo como *características da técnica de implementação*?	Sim ( ) Não ( ) N.A. ( )
10	As características que não possuem ligações concretas com o domínio, mas facilitam o seu entendimento, estão representadas no modelo como *características organizacionais*?	Sim ( ) Não ( ) N.A. ( )
11	Alguma característica do modelo, embora correta, está fora do escopo do modelo, não contribuindo para o entendimento do domínio?	Sim ( ) Não ( ) N.A. ( )
12	Existem características distintas no modelo que representam um mesmo conceito do domínio?	Sim ( ) Não ( ) N.A. ( )

continuação na próxima página

13	Algum conceito relevante do domínio deixou de ser incluído no modelo?	Sim ( ) Não ( ) N.A. ( )

Fonte: Mello et al. (2014).

Os dezesseis itens do grupo "Verificação dos relacionamentos entre as características do modelo" são apresentados na Tabela 44.

Tabela 44 - Itens para verificação dos relacionamentos entre as características do modelo.

#	**Item de verificação**	**Resposta**
14	As situações do domínio em que uma ou mais de uma característica podem ser escolhidas dentro de um conjunto de características (OU, OU Exclusivo) estão devidamente representadas no modelo?	Sim ( ) Não ( ) N.A. ( )
15	As cardinalidades dos pontos de variação do modelo estão corretas?	Sim ( ) Não ( ) N.A. ( )
16	Os pontos de variação do modelo estão descritos com clareza e suas descrições refletem as características que os compõem?	Sim ( ) Não ( ) N.A. ( )
17	Duas ou mais características do modelo estão reunidas em um relacionamento, mas não é possível identificar este relacionamento no domínio?	Sim ( ) Não ( ) N.A. ( )
18	Existe algum relacionamento entre características que deixou de ser informado no modelo?	Sim ( ) Não ( ) N.A. ( )
19	A hierarquia entre as características está em conformidade com o domínio?	Sim ( ) Não ( ) N.A. ( )
20	Alguma característica foi indevidamente apontada como generalização de outra?	Sim ( ) Não ( ) N.A. ( )
21	As características que estão apontadas no modelo como "implementada por" outra característica, possuem este relacionamento no domínio?	Sim ( ) Não ( ) N.A. ( )

continuação na próxima página

22	Os relacionamentos de agregação e de composição entre características do domínio apontados no modelo condizem com a realidade deste domínio?	Sim ( ) Não ( ) N.A. ( )
23	Alguma dependência ou relação de mútua exclusividade entre características no modelo não se aplica ao domínio descrito?	Sim ( ) Não ( ) N.A. ( )
24	Alguma dependência ou relação de mútua exclusividade entre características deixou de ser informada no modelo?	Sim ( ) Não ( ) N.A. ( )
25	A presença de alguma característica no modelo contraria outra característica do mesmo modelo?	Sim ( ) Não ( ) N.A. ( )
26	A característica-raiz auxilia a compreensão do domínio que ela e as demais características do modelo buscam descrever?	Sim ( ) Não ( ) N.A. ( )
27	De um modo geral, é possível compreender o domínio a partir de suas características?	Sim ( ) Não ( ) N.A. ( )
28	O modelo descreve o domínio em um nível de detalhamento adequado para ser compreendido sob a perspectiva pretendida?	Sim ( ) Não ( ) N.A. ( )
29	O modelo apresenta as características necessárias e suficientes para orientar sua implementação?	Sim ( ) Não ( ) N.A. ( )

Fonte: Mello et al. (2014).

Os cinco itens do grupo "Verificação das regras de composição do modelo" são apresentados na Tabela 45.

Tabela 45 - Itens para verificação das regras de composição entre as características do modelo.

#	**Item de verificação**	**Resposta**
30	Todas as regras de composição do modelo estão descritas com clareza e objetividade e estão em conformidade com o domínio?	Sim ( ) Não ( ) N.A. ( )

continuação na próxima página

31	Alguma regra de composição do modelo contraria outra regra de composição do mesmo modelo?	Sim ( ) Não ( ) N.A. ( )
32	Alguma regra de composição do modelo não se aplica ao domínio, embora possa estar correta?	Sim ( ) Não ( ) N.A. ( )
33	Todas as regras de composição necessárias para descrever o domínio foram devidamente representadas no modelo?	Sim ( ) Não ( ) N.A. ( )
34	O modelo apresenta as regras de composição necessárias e suficientes para orientar sua implementação?	Sim ( ) Não ( ) N.A. ( )

Fonte: Mello et al. (2014).