UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Guidelines para adoção do Event Storming com foco em Arquitetura de Microsserviços

Dissertação de Mestrado

Marcos Cesar Barbosa dos Santos

São Cristóvão – Sergipe

2025

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcos Cesar Barbosa dos Santos

Guidelines para adoção do Event Storming com foco em Arquitetura de Microsserviços

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de mestre em Ciência da Computação.

Orientador(a): Dr. Fábio Gomes Rocha

São Cristóvão – Sergipe

2025

FICHA CATALOGRÁFICA ELABORADA PELO SIBIUFS
UNIVERSIDADE FEDERAL DE SERGIPE

S237g

Santos, Marcos Cesar Barbosa dos
Guidelines para adoção do Event Storming com foco em arquitetura de microsserviços / Marcos Cesar Barbosa dos Santos ; orientador Fábio Gomes Rocha. – São Cristóvão, SE, 2025.
72 f. : il.

Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Sergipe, 2025.

1. Computação. 2. Software – Desenvolvimento. 3. Programação orientada a objetos (Computação) – Diretrizes. 4. UML (Computação). I. Rocha, Fábio Gomes, orient. II. Título.

CDU 004.4:005.53

UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
COORDENAÇÃO DE PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ata da Sessão Solene de Defesa da Dissertação do Curso de Mestrado em Ciência da Computação-UFS.
Candidato: Marcos Cesar Barbosa dos Santos

Em 28 dias do mês de Agosto do ano de dois mil e vinte cinco, com início às 18hs, realizou-se na Sala de Seminários do PROCC da Universidade Federal de Sergipe, na Cidade Universitária Prof. José Aloísio de Campos, a Sessão Pública de Defesa de Dissertação de Mestrado do candidato **Marcos Cesar Barbosa dos Santos**, que desenvolveu o trabalho intitulado: **" Guidelines para adoção do Event Storming com foco em Arquitetura de Microsserviços",** sob a orientação do Prof. Dr. **Fábio Gomes Rocha.** A Sessão foi presidida pelo Prof. Dr. **Fábio Gomes Rocha** (PROCC/UFS), que após a apresentação da dissertação passou a palavra aos outros membros da Banca Examinadora, Prof. Dr. **Guillermo H. Rodríguez** (UBA) e, em seguida, o Prof. Dr. **Gilton José Ferreira da Silva** (Procc/UFS). Após as discussões, a Banca Examinadora reuniu-se e considerou o mestrando (a) Aprovado "*(aprovado/reprovado)"*. Atendidas as exigências da Instrução Normativa 05/2019/PROCC, do Regimento Interno do PROCC (Resolução 67/2014/CONEPE), e da Resolução nº 04/2021/CONEPE que regulamentam a Apresentação e Defesa de Dissertação, e nada mais havendo a tratar, a Banca Examinadora elaborou esta Ata que será assinada pelos seus membros e pelo mestrando.

Cidade Universitária "*Prof. José Aloísio de Campos",* 28 de Agosto de 2025.

gov.br Documento assinado digitalmente
FABIO GOMES ROCHA
Data: 10/09/2025 08:58:15-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fábio Gomes Rocha
(PROCC/UFS)
Presidente

Prof. Dr Guillermo H. Rodríguez
(UNICEN)
Examinador Externo

gov.br Documento assinado digitalmente
GILTON JOSE FERREIRA DA SILVA
Data: 08/09/2025 19:59:33-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gilton José Ferreira da Silva
(PROCC/UFS)
Examinador Interno

gov.br Documento assinado digitalmente
MARCOS CESAR BARBOSA DOS SANTOS
Data: 09/09/2025 09:14:30-0300
Verifique em https://validar.iti.gov.br

Marcos Cesar Barbosa dos Santos
Candidato

Dedico este trabalho a todas as pessoas que, assim como eu, precisaram conciliar os desafios da vida com as exigências do mestrado. A cada um que seguiu em frente mesmo diante do cansaço, da dúvida e da rotina exaustiva, fica aqui a minha mensagem de perseverança.

Agradecimentos

Primeiramente eu gostaria de agradecer à todos aqueles que me incentivaram ou prestaram algum suporte quando decidi me inscrever no mestrado. Em especial gostaria de agradecer à Raquel que além de me incentivar me proporcionou uma oportunidade de lecionar. Também gostaria de agradecer à Flavia que me acolheu em sua pesquisa quando eu era apenas um pesquisador iniciante. Também agradeço à Shexmo que nunca larga minha mão durante os surtos que nos acometem entre a escrita de um artigo e outro. Reservo também meus agradecimentos ao meu orientador Fábio por exercer esse papel importante na condução de minha pesquisa. E por fim, agradeço também ao Tribunal de Jusitiça do Estado de Sergipe.

Resumo

Times ágeis de desenvolvimento necessitam de definições de requisitos bem explícitas para manter a produtividade. O *Event Storming* se apresenta como uma técnica dinâmica para produção desse tipo de documentação visual com baixos formalismos e compatível com a arquitetura de microsserviços. Por ser uma técnica relativamente recente e com poucas regras quando comparada à documentação mais estabelecida, como a UML, tanto equipes ágeis quanto gestores podem demonstrar resistência em utilizar o *Event Storming*. Como uma demonstração de eficácia, a construção de diretrizes para a adoção do *Event Storming* durante a fase de requisitos facilitará tanto a condução da técnica por novos times de desenvolvimento quanto aumentará a confiabilidade de toda a abordagem, diminuindo potencialmente a resistência enfrentada pelas equipes de desenvolvimento e gestão. Com o objetivo de construir conhecimento a partir do design de artefatos experimentais e evolutivos, esta pesquisa se baseia na metodologia do *Design Science Research*. Espera-se que as *guidelines* propostas a partir desse estudo promovam maior clareza na elicitação de requisitos e se demonstrem eficientes no sentido de favorecer a integração do *Event Storming* com outras práticas ágeis, especialmente em contextos de microsserviços.

Palavras-chave: *Event storming*, Requisitos, UML, Times ágeis

Abstract

Context: Agile development teams require well-defined requirements to maintain productivity. Event Storming serves as a dynamic technique for producing this type of visual documentation with low formalism and compatible with the microservices architecture. Problem: As it is a relatively recent technique with few rules when compared to more established documentation, such as UML, both agile teams and managers may show resistance in using Event Storming. Solution: As a proof of effectiveness, the construction of guidelines for the adoption of Event Storming during the requirements phase will facilitate both the conduct of the technique by new development teams and will increase the reliability of the entire approach, potentially reducing the resistance faced by development and management teams. Method: With the aim of building knowledge from the design of experimental and evolutionary artifacts, this research is based on the methodology of Design Science Research.

Keywords: event storming, requirements, UML, agile teams.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
BDD	Behavior-Driven Development
DCOMP	Departamento de Computação
DDD	Domain-Driven Design
DFD	Data Flow Diagram
ES	Event Storming
LLM	Large Language Model
MVP	Minimum Viable Product
UFS	Universidade Federal de Sergipe
UML	Unified Modeling Language

Sumário

1 Introdução

A arquitetura de microsserviços tem sido amplamente adotada nas organizações, tornando-se fundamental para o desenvolvimento moderno de software (RODRIGUEZ et al., 2023). Essa abordagem permite a divisão de sistemas monolíticos em serviços menores e interconectados, facilitando sua manutenção, escalabilidade, tolerância a falhas e integração, que são requisitos cruciais em diversos setores. No entanto, o aumento na quantidade de serviços traz consigo uma considerável complexidade quando comparado aos sistemas de arquitetura monolítica (CHEN, 2019). Assim, esse estilo arquitetural gera novas demandas ligadas ao *design*, devido à segmentação de componentes, o que resulta em necessidades de comunicação orientada a eventos, especialmente comunicação assíncrona (WEERASINGHE; PERERA, 2023), requerendo uma análise e modelagem aprimoradas para evitar problemas futuros.

Embora existam diversas abordagens e metodologias focadas na modelagem de serviços, poucas foram amplamente aplicadas no contexto de microsserviços. Nesse cenário, o *Event Storming* (BRANDOLINI, 2018) se apresenta como uma técnica de modelagem colaborativa que auxilia as equipes a identificar, entender e visualizar o fluxo dos vários eventos em um sistema de *software*. Essa técnica permite que as partes interessadas, desenvolvedores e outros membros da equipe criem um modelo real e visual que represente o domínio do sistema. O *Event Storming* utiliza componentes que permitem a identificação de comportamentos e funcionalidades, resultando na descoberta de requisitos funcionais, bem como demandas de comunicação e integração, gerando também requisitos não funcionais.

Dessa forma, o *design* de sistemas na arquitetura de microsserviços pode se beneficiar significativamente de metodologias como o *Event Storming*, que possibilita a identificação dos limites de cada domínio e subdomínio da aplicação, permitindo a criação de sistemas sustentáveis (JUNKER, 2021). Como os microsserviços são dirigidos a domínio (EVANS, 2004), seu *design* precisa passar por uma análise de domínio que irá definir os limites de cada microsserviço. A representação desses domínios pode ser feita utilizando diagramas onde é possível representar

cada microsserviço e seus respectivos domínios, assim como as interações entre eles. O diagrama resultante do *Event Storming* (*big picture*) atende a esta necessidade, sendo uma forte candidata a ferramenta de representação de domínio.

No contexto do *design* de microsserviços, em times ágeis, a documentação pode ser negligenciada, desatualizada ou apenas ignorada ao longo do processo de desenvolvimento (ROCHA; MISRA; SOARES, 2023). As notações e linguagens para se viabilizar essa documentação costumam ser verbosas ou com regras extensas, e isso contribui para que esta documentação dos microsserviços não receba a devida atenção. Para este desafio, o *Event Storming* fornece uma solução de documentação interessante, com poucas regras e visualmente coesa.

A própria modelagem de sistemas apresenta desafios quanto à complexidade de sistemas que exigem a construção de modelos mais detalhados. Estes modelos ainda precisam se adaptar aos eventos que podem ocorrer durante o processo de desenvolvimento, como a mudança de requisitos provocada tanto por má definição quanto por mudanças provocadas por agentes externos.

1.1 Objetivos

Diante do exposto, o objetivo deste estudo é avaliar a aplicabilidade do *Event Storming* na definição de requisitos funcionais e não funcionais, dentro do contexto de uma arquitetura de microsserviços. Este método visa produzir um diagrama organizado e de fácil entendimento, facilitando a visualização e a organização dos eventos que compõem o sistema. Além disso, os procedimentos necessários para a elaboração deste diagrama serão estruturados com o intuito de garantir resultados mais eficientes e eficazes.

Ao final deste estudo, serão apresentadas diretrizes para a adoção do *Event Storming* no desenvolvimento de sistemas baseados em microsserviços, contribuindo para uma melhor prática de modelagem e implementação. Considerando este o objetivo geral, com o intuito de estruturar a pesquisa seguem os objetivos específicos:

OE1: Avaliar a capacidade de utilização do *Event Storming* como ferramenta de análise de requisitos.

OE2: Produzir *guidelines* para aplicação do *Event Storming* como ferramenta de análise de requisitos.

OE3: Avaliar a eficácia das *guidelines* a partir de uma aplicação técnica apoiada com Inteligência Artificial

Com base nos objetivos definidos, esta pesquisa busca responder às seguintes perguntas de pesquisa:

P1: Como o *Event Storming* pode ser utilizado de forma eficaz como ferramenta para análise de requisitos em contextos de times ágeis?

P2: Quais *guidelines* podem ser formuladas para orientar a aplicação do *Event Storming* como técnica de elicitação e análise de requisitos?

P3: As *guidelines* propostas contribuem efetivamente para a aplicação do *Event Storming* com suporte de inteligência artificial generativa na geração de requisitos de *software*?

Uma vez que o *Event Storming* é uma técnica relativamente recente, ainda em processo de consolidação no meio acadêmico, mas já utilizada em ambientes corporativos, torna-se relevante a produção de conhecimento que agregue valor ao estado da arte na área de Engenharia de Requisitos. Neste sentido, escolher a metodologia científica apropriada é fundamental para garantir a qualidade deste conhecimento criado. Assim adotou-se o *Design Science Research* (DSR) como metodologia por sua natureza investigativa, resultando na construção e avaliação de artefatos inovadores voltados à solução de problemas práticos e à geração de conhecimento aplicável.

Nas próximas seções, serão apresentados o conceito teórico e as metodologias empregadas no processo.

2

Referencial Terórico

O referencial teórico deste trabalho está dividido em três abordagens principais. A primeira abordagem, *Unified Modeling Language* (UML) e *Systems Modeling Language* (SysML), é amplamente utilizada para a modelagem de sistemas complexos, oferecendo uma série de diagramas e técnicas que ajudam a capturar e comunicar o comportamento e a estrutura dos sistemas. A segunda abordagem é *Lean Inception*, que contribui para a delimitação de um escopo essencial, permitindo assim que os trabalhos não sejam demasiado longos ou complexos. Por fim, o *Event Storming* é uma técnica de modelagem colaborativa que se concentra na identificação e visualização dos eventos de negócios de um sistema. Este método tem se mostrado eficaz na captura de requisitos de alto nível e na promoção de um entendimento compartilhado entre as partes interessadas, facilitando a transição para uma arquitetura de microsserviços. Além dessas abordagens, o referencial teórico também incluirá uma seção dedicada à arquitetura de microsserviços, abordando suas características, e uma seção contendo trabalhos relacionados ao tema.

2.1 UML e SysML

O *design* de software para microsserviços utilizando UML e SysML enfrenta diversos desafios, incluindo a exposição de serviços, a comunicação entre serviços e a implantação de infraestrutura (SANTOS et al., 2019). Abordagens orientadas por modelos oferecem uma abstração do comportamento dos microsserviços em relação ao domínio comercial; contudo, ainda há uma demanda significativa por métodos que abordem esses desafios específicos de maneira eficaz (SANTOS et al., 2019).

A extensão dos elementos da linguagem UML, juntamente com o uso de diagramas SoaML, permite que arquiteturas lógicas de microsserviços sejam derivadas a partir de requisitos funcionais, alinhando o *design* às necessidades de negócios e integrando os princípios fundamen-

tais dos microsserviços (CARRANZA-GARCÍA; RODRÍGUEZ-DOMÍNGUEZ; GARRIDO, 2021; PETRASCH, 2017). Além disso, a aplicação de Padrões de Integração Empresarial (EIP) no *design* de microsserviços aprimora a integração de serviços por meio de mensagens e tratamento de eventos, oferecendo uma especificação precisa de microsserviços para simulações, transformações ou tarefas de geração de código (PETRASCH, 2017). Essas metodologias não apenas contribuem para a automatização do desenvolvimento de software, mas também melhoram a escalabilidade e o *design* de sistemas distribuídos em ambientes onipresentes (CARRANZA-GARCÍA; RODRÍGUEZ-DOMÍNGUEZ; GARRIDO, 2021).

Apesar de a utilização de UML e SysML proporcionar uma base de modelagem de microsserviços, permitindo uma melhor compreensão e comunicação das complexidades inerentes a esses sistemas (VICTOROVA; OLEYNIK, 2024), sua aplicação prática enfrenta algumas dificuldades. A linguagem UML oferece uma série de diagramas para a modelagem de software; entretanto, ela se mostra uma abordagem verbosa e potencialmente lenta. Para a determinação de casos de uso, estima-se que apenas 10% deles sejam completos, enquanto o restante deve ser informal ou simples (LARMAN et al., 1998).

Adicionalmente, torna-se evidente a necessidade de ferramentas de apoio para otimizar sua utilização, dada a complexidade inerente à UML, que pode ser intimidante ou resistente àqueles não familiarizados com sua sintaxe e diagramas (RUMPE, 2002). A construção desses diagramas segue uma série de regras que, mesmo sendo expansíveis, tornam o processo de produção e atualização lento, por vezes, negligenciado.

A linguagem UML oferece uma série de diagramas para a modelagem de software, entretanto ela se mostra uma abordagem verbosa e potencialmente lenta. Para a determinação de casos de uso, estima-se que 10% deles sejam completos enquanto o restante deve ser informal ou simples (LARMAN et al., 1998).

No entanto, torna-se evidente a necessidade de ferramentas de apoio para otimizar sua utilização, dada a complexidade inerente à UML, que pode ser intimidante ou resistente àqueles não familiarizados com sua sintaxe e diagramas (RUMPE, 2002).

A construção destes diagramas segue uma série de regras que, mesmo sendo expansíveis, tornam o processo de produção e atualização lento e por vezes negligenciado.

2.2 Lean Inception

Lean Inception é uma metodologia em formato de *workshop* criada por Paulo Caroli e tem como objetivo a criação de um produto mínimo viável ao qual chamamos de *Minimum Viable Product* (MVP). O foco é gerar um produto alinhado com a equipe de negócio, a equipe técnica e o usuário final, evitando o desperdício (CAROLI, 2018).

Uma sessão de *Lean Inception* originalmente é conduzida ao longo de uma semana, e

suas etapas ocorrem sequencialmente. Cada etapa visa focar em um aspecto do projeto e estão listadas a seguir:

- Alinhamento sobre o propósito – As atividades neste momento são de descobrir o objetivo macro na visão dos participantes de como é o produto. É o primeiro momento em que as expectativas são alinhadas e as definições ainda são amplas.
- Visão do produto – Ainda alinhando as expectativas, mais objetivos são definidos e descartados, unificando ainda mais a visão de produto.
- Personas – Utilizando abordagem de *Design Thinking* (SIRICHAROEN, 2020) as personas são criadas a fim de representar os possíveis usuários com seus mais variados objetivos e limitações.
- Jornada do usuário – As atividades dos usuários são ilustradas com post-its. Dessa maneira, é possível identificar os pontos de interação dos usuários com o sistema.
- *Brainstorm* de funcionalidades – A partir das personas e objetivos, as funcionalidades do sistema ganham definição.
- Avaliação de complexidade, valor e *User Experience* (UX) - Etapa onde os participantes qualificam o grau de esforço técnico para construir a funcionalidade, a importância para o negócio e a satisfação do usuário ou persona.
- Sequenciador de funcionalidades – Priorização das funcionalidades com base em valor de negócio, satisfação de usuário e complexidade. É onde se define o que será entregue em cada versão.
- Canvas do MVP – Uma organização visual apresentável que resume o potencial do produto a ser gerado.

2.3 Event Storming

Existem diversas técnicas que auxiliam na engenharia de requisitos. As técnicas onde há interação entre membros da equipe de desenvolvimento e equipe de negócio são os maiores exemplos. O *Event Storming* (BRANDOLINI, 2018), por exemplo, é uma técnica baseada em *workshop* que usa post-its para construir uma narrativa.

Assim, exploram-se rapidamente domínios de negócios complexos de maneira eficiente, viabilizando o entendimento do sistema por meio da elaboração de questionamentos capazes de produzir um resultado. Durante cada etapa, o modelo é enriquecido com informações que proporcionam um ambiente colaborativo bem como um conhecimento mais profundo do sistema, facilitando *insights* sobre os requisitos e restrições do sistema.

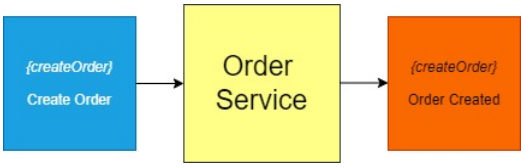
Cada post-it desempenha uma função de acordo com a sua cor. E a partir da interação entre eles é possível traçar os possíveis pontos de interação do *software*. Os eventos são o início do *workshop*. Portanto, são representados por um post-it laranja. O seu texto foi estruturado a partir de um sujeito e um verbo no passado, com o objetivo de traduzir uma visão clara da parte comportamental do sistema, possibilitando compreender o domínio e as complexidades do sistema existente.

Figura 1 – Eventos Criados de Forma Aleatória

Fonte: Elaborado pelo autor.

As dúvidas estão representadas por post-its rosa e não há regra para nomeá-los. Podem representar as dúvidas, preocupações, problemas ou até mesmo perguntas. Os comandos que são criados a partir dos eventos, sendo, portanto, responsáveis pelo início de um evento. A cor do post-it é azul e seus nomes são dados de acordo com o evento que se conecta, com o verbo no infinitivo.

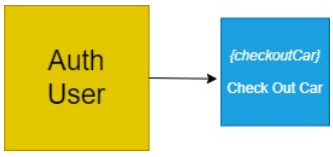
Figura 2 – Um Comando e Seus Eventos de Domínio Resultantes



Fonte: Elaborado pelo autor.

Atores representam os usuários que interagiram com o sistema por meio dos comandos. Estão representados pelos post-its amarelos. Uma vez estabelecido o papel do ator, é possível visualizar quem, como e quando interage com o sistema. Essa visualização é essencial para entender bem como modularizar o sistema que reflita o negócio.

Figura 3 – Exemplo de Ator Executando Comando

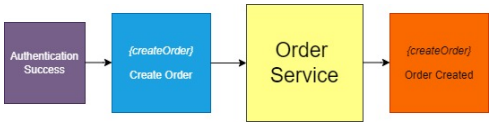


Fonte: Elaborado pelo autor.

As políticas correspondem aos post-its lilás; sua função é representar uma condição ou um grupo de condições iniciadas por eventos para executar um comando específico.

Agregadores reúnem os comandos e eventos registrados conectando seus contextos. A representação é por um post-it amarelo-claro e ajuda a estabelecer *bounded contexts* dentro do

Figura 4 – Exemplo de Política Aplicada a Um Comando



Fonte: Elaborado pelo autor.

sistema, permitindo esforços de refatoração mais focados e gerenciáveis, diminuindo drasticamente a carga cognitiva dos times.

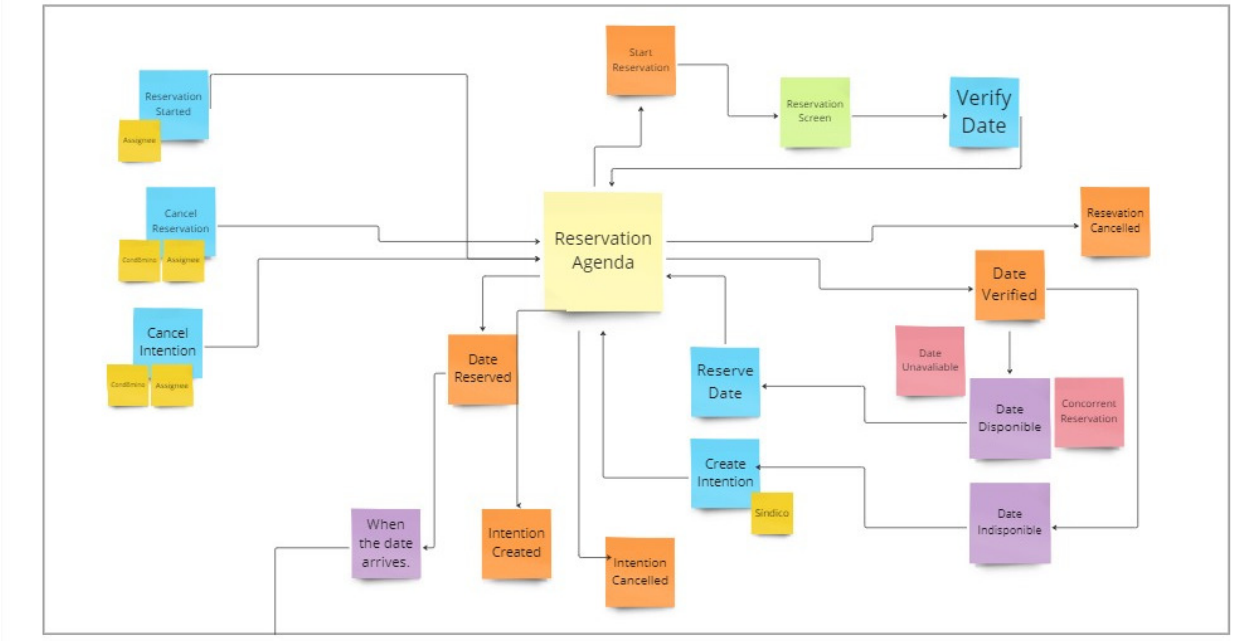
Fazendo uma alusão ao Princípio de Pareto (LAOYAN, 2022), a menor parte dos recursos é responsável pela maior parte dos resultados, ou seja, o esforço concentrado em poucas ações vai efetivamente gerar resultados significativos.

2.3.1 Artefatos do *Event Storming*

O principal artefato gerado pelo *Event Storming* é a *Big Picture*, uma documentação viva que reúne os comandos e eventos necessários para representar o objeto do workshop.

Quando todos os post-its estiverem organizados, é possível observar todos os contextos interagindo uns com os outros em uma grande figura 5.

Figura 5 – Exemplo Big Picture do Event Storming



Fonte: Elaborado pelo autor.

Por ser feita em linguagem ubiqua, a *big picture* é de fácil entendimento uma vez que seus elementos representativos são cartões coloridos e linhas. A facilidade de se alterar esse conjunto de cartões permite uma versatilidade maior em relação aos diagramas da UML.

É importante ressaltar que a *big picture* ou qualquer outro artefato gerado numa sessão de *Event Storming* não visa substituir a UML ou qualquer outra linguagem ou notação de modelagem. Tanto que o *Event Storming* apresenta-se compatível com os diagramas de casos de uso e os diagramas de atividade e sequência, sendo simples o suficiente para atuar como complemento da modelagem.

2.3.2 Linguagem ubíqua e sua importância no desenvolvimento de *software*

Uma linguagem ubíqua é uma abstração mínima em que é possível representar informações com o mínimo de explicação prévia. Utiliza símbolos e iconografia para comunicar suas informações e é a base de metodologias como o *Domain Storytelling* (HOFER; SCHWENTNER, 2021) e o *Domain-Driven Design* (DDD) (VERNON, 2013).

A utilização de uma linguagem ubíqua permite a participação dos *stakeholders* externos, o que possibilita uma maior confiabilidade no resultado final do processo.

Migrar os sistemas monolíticos para uma arquitetura baseada em Microserviços é uma realidade enfrentada por diversas equipes e até existem estratégias para que essa transição ocorra (LAURETIS, 2019). O *Event Storming* pode ser útil para que, durante o processo de engenharia reversa, a criação da *big picture* permita a identificação dos *Bounded Contexts*, sendo assim viável a elicitação de quais áreas serão isoladas para serem substituídas por microsserviços (NEWMAN, 2019).

2.4 Microserviços

A arquitetura de microsserviços é um paradigma arquitetural que emerge da arquitetura orientada a serviços (SOA) (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016; ROCHA; SOARES; RODRIGUEZ, 2023) e que vem se tornando popular na indústria de desenvolvimento de software nas últimas décadas (ROCHA; SOARES; RODRIGUEZ, 2023). Em uma arquitetura de microsserviços, uma aplicação é desmembrada em pequenos serviços independentes, cada um executando funções específicas. Cada serviço é construído, implantado e dimensionado de forma independente. Essa abordagem oferece maior flexibilidade, escalabilidade e manutenção em comparação com a arquitetura monolítica (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016; CHRISTOFOROU et al., 2017). Desta forma, para se ter serviços independentes, coesos e que permitem colaboração, pode-se usar o DDD para definir o escopo funcional dos microsserviços (MERSON; YODER, 2020; SAIDI; TISSAOUI; FAIZ, 2023).

O DDD é uma abordagem sistêmica que permite a modelagem de domínio de problemas (EVANS, 2004), ligando o design ao desenvolvimento de software, tendo como principal foco a redução da complexidade. O DDD fornece um conjunto de ferramentas, técnicas e padrões que permite identificar e modelar sistemas que auxiliam em projetos, podendo ser aplicado de

forma a delimitar os serviços, decompondo-os em diversos aplicativos implementáveis, ou seja, microsserviços (KAPFERER; ZIMMERMANN, 2020).

Ao segmentar os serviços em aplicações independentes, surge um novo problema, relacionado à superfície de ataque, pois há diversos pontos de entrada que podem ser utilizados para ataques (CHEN, 2019). Uma superfície de ataque é o conjunto de todos os pontos de entrada que podem ser explorados por um atacante para comprometer a segurança de um sistema, ou seja, está ligado a pontos críticos de um sistema que são acessíveis externamente (MOSHTARI; OKUTAN; MIRAKHORLI, 2022). Em arquiteturas de microsserviços, a Superfície de Ataques é ampliada devido à natureza distribuída do sistema. Isso inclui APIs expostas, interfaces de comunicação entre serviços e componentes externos. Assim, uma etapa importante para a construção de microsserviços é a análise de superfície de ataque.

A análise da superfície de ataque tem como objetivo a identificação de recursos passíveis de ataques, modelando a superfície de ataque e analisando operações e comportamentos, fornecendo assim uma visão sobre resultados de medição relacionados à segurança de um sistema de informações e as possíveis recomendações (CHRISTOFOROU et al., 2017). Assim, para realizar a análise de superfície de ataque, emprega-se uma modelagem; em geral, o DDD é empregado por possibilitar identificar os diversos componentes e interações no sistema, permitindo analisar a superfície passível de ataque (BELLOVIN, 2016).

A modelagem de uma aplicação para a análise de superfície de ataques tem como objetivo a elaboração de modelos que permitam analisar o sistema e como as pessoas o utilizarão, de forma a permitir a implementação de práticas de segurança, minimizando pontos de falhas de segurança. Porém, o DDD não é um modelo amplamente adotado em contextos de microsserviços ou da adoção do DDD, surge assim o *Event Storming*, proposto por Brandolini, sendo uma técnica de colaboração que auxilia equipes a compreender a lógica do negócio e os fluxos dos eventos em um domínio (BRANDOLINI, 2018). Sendo uma abordagem visual, esta envolve a criação de um modelo de eventos de domínio, tornando-se assim uma forma valiosa para a análise de superfície de ataque, considerando que durante a execução do *Event storming* são identificados os usuários, os comandos e eventos disparados, além de possuir os fluxos entre os componentes, permitindo identificar possíveis alvos de ataque.

Apesar dos benefícios que o *Event Storming* pode promover na definição de requisitos de software, é importante salientar que a proposta de sua utilização não visa substituir técnicas, métodos ou ferramentas já consolidadas como a UML, BPMN e DFD. Ao longo do desenvolvimento da pesquisa, o *Event Storming* eventualmente será comparado a essas ferramentas apenas para medir seus resultados. Não é de interesse dessa pesquisa substituir UML, BPMN ou DFD no processo de levantamento de requisitos, e sim de complementar e evoluir o uso dessas ferramentas.

3

Trabalhos Relacionados

Este trabalho emerge de (ROCHA; MISRA; SOARES, 2023) que se preocupa em criar *Guidelines* para contextos ágeis em uma perspectiva futura envolvendo arquitetura de software e metodologias ágeis. Com o objetivo de encontrar um espaço para o *Event Storming* na análise de requisitos, foram separados e categorizados os trabalhos que de algum modo podem guiar a pesquisa e apresentar as lacunas necessárias para justificar a criação das *guidelines*. Os trabalhos foram organizados em 3 grupos : técnicas visuais para análise de requisitos; análise de segurança nas etapas de *design* e; uso de IA no contexto da Engenharia de Software

Como o *Event Storming* é uma técnica recente, poucos estudos acadêmicos foram conduzidos até a realização desta dissertação. Ao se pesquisar nos repositórios científicos do CAPES e Scopus, poucos resultados foram encontrados e destes mesmos resultados o uso da técnica não era analisado em si.

3.1 Técnicas visuais para análise de requisitos

Ao pesquisar artigos relacionados que abordem a técnica *Event Storming*, poucos trabalhos são obtidos. Sendo escassos os resultados, os critérios para seleção de trabalhos relacionados acaba por ser a abordagem do assunto no âmbito da engenharia de software. Os artigos mais relevantes são expostos no Quadro 2.

Uludağ et al. (2018) expõem suas preocupações quando o *Domain-Driven Design* é usado para projetos grandes em times ágeis. Eles propõem o uso do *Event Storming* e de um *framework* para guiar o processo de levantamento de requisitos. Os desafios enfrentados por grandes projetos que utilizam de metodologia ágil (EDISON; WANG; CONBOY, 2021) estão relacionados no Quadro 1

Wojczal (2023) em seu artigo no Linkedin, define um guia para executar entregas rápidas utilizando o *Event Storming* acoplado a uma metodologia ágil. Seu guia estabelece uma série de

Quadro 1 – Principais desafios na utilização de métodos ágeis em projetos de grande porte

Relação de desafios encontrados	
P	**Problema**
P1	Coordenação inter-time
P2	Desafios de estrutura organizacional
P3	Desafios de arquitetura
P4	Desafios de engenharia de requisitos
P5	Desafios de colaboração de cliente
P6	Desafios de adoção de métodos relacionados
P7	Desafios de mudança de gestão
P8	Desafios relacionados à equipe
P9	Desafios de gerenciamento de projeto

Fonte: Elaborado pelo autor.

ações de planejamento incluindo várias sessões de *Event Storming* e versões de testes semanais. Ele também destaca a importância da linguagem ubíqua para se obter resultados satisfatórios e a adaptação de ferramentas existentes para manter a essência do ágil

Quadro 2 – Trabalhos Relacionados — Técnicas Visuais para Engenharia de Requisitos

Referência	**Contribuições**	**Limitações**	**Lacunas Exploradas na Dissertação**
Brandolini (2018)	Base do Event Storming: linguagem ubíqua, eventos, comandos, agregados; condução de workshops.	Pouca validação formal; sem integração com IA.	Guidelines operacionais; validação empírica; indicação de uso de LLM em prompts e artefatos.
Uludağ et al. (2018)	Event Storming com DDD para requisitos em projetos ágeis de grande porte.	Foco em larga escala; não detalha IA/segurança desde o design.	Ampliar aplicabilidade a contextos diversos; integrar segurança e apoio de LLM.
Wojczal (2023)	Guia prático para planejar sessões de ES e ciclos iterativos de entrega.	Material não acadêmico; sem avaliação empírica; foco formal em requisitos limitado.	Formalizar boas práticas como diretrizes; validar empiricamente no estudo/experimento.
Rocha, Misra e Soares (2023)	Uso de guidelines em contextos ágeis e reconciliação com arquitetura.	Não aborda ES nem suporte de LLM diretamente.	Adotar o formato "guidelines" direcionado à prática de ES com IA na engenharia de requisitos.
Edison, Wang e Conboy (2021)	Mapeamento de desafios da Engenharia de Requisitos Ágil.	Análise ampla e não prescritiva; sem artefato operacional.	Converter achados em diretrizes práticas e replicáveis para condução de ES.

Fonte: Elaborado pelo autor.

3.2 Análise de segurança nas etapas de *design*

Em seu artigo, Zhang (2023) debate sobre a solução IAST através da proposição de estudo comportamental fundamentada em aprendizado de máquina. Esse aprendizado se desenvolve por meio da localização e da exploração de vulnerabilidades em tempo real segundo as variadas que estão embasadas em microserviços garantidores de segurança. Assim, a IAST visa instituir compatibilidade com segurança usando políticas impulsionadas para detectar explorações de vulnerabilidade, falsos positivos e detecção de anomalias. A ideia preconizada é construir a confiança aumentando a probabilidade e, por fim, implantar o modelo para monitorar o status de segurança em tempo real.

Reselman (2022a), Reselman (2022b), Reselman (2022c) inicia sua série expondo cinco princípios de *design* para microsserviços. Os princípios em questão (RESELMAN, 2022a) se referem às responsabilidades do microsserviço (cada microsserviço possui apenas uma responsabilidade), limites discretos (o microsserviço é encapsulado e separado de seu ambiente), transportabilidade (microsserviços podem ser transferidos para ambientes diferentes com pouco esforço), domínio próprio (carregam seus próprios dados) e efemeridade (os microsserviços são inerentemente efêmeros, sendo criados e destruídos conforme a necessidade da aplicação).

Uma revisão sistemática de Rezapour et al. (2021) apresenta os conceitos necessários em *fog computing* e os requisitos para adquirir o conceito de segurança nesta área.

O estudo apresentado por Roman, Lopez e Mambo (2018) procura fornecer, de uma perspectiva holística, uma análise detalhada da segurança dos paradigmas de borda, por meio da análise de uma perspectiva de ameaças e desafios de segurança que afetam os paradigmas como: computação em névoa, computação de borda móvel e computação em nuvem móvel.

Conforme Levcovitz, Terra e Valente (2016) propõe uma técnica para identificar e definir serviços em um ambiente monolítico empresarial, considerando a ideia de que um grande sistema é estruturado em subsistemas menores, a técnica considera que aplicações monolíticas possuem três partes principais: Interface de usuário do lado do cliente, um aplicativo do lado do servidor e Banco de Dados. Estruturado em subsistemas menores, com cada subsistema possuindo um conjunto de responsabilidades, é possível identificar os "candidatos"
a ser transformados em microsserviços. Assim, a arquitetura monolítica é considerada eficaz até que se converta em desafio, tornando-se muito grande e com alta escalabilidade. Isto acontece porque é um único processo executado em toda a base de código. Diferentemente dos microsserviços que são caracterizados pela descentralidade, afinal desacoplam em pequenos serviços para que o processamento funcione separadamente.

Uma abordagem de modelagem e análise de segurança para sistemas orientados a objetos com a adoção de análise de fluxo de informações foi proposta por Herrmann (2001), que adotou técnicas de modelagem baseadas em UML para a análise da superfície de ataques.

Já Jürjens, Schreck e Bartmann (2008) propuseram um método composto de uma

ferramenta denominada UMLSec que envolve a modelagem dos requisitos de segurança e a realização da análise de segurança de aplicações.

O trabalho de Georg et al. (2010) apresentou um meio de analisar a segurança de aplicação por meio da metodologia *Aspect-Oriented Risk-Driven Development* (AORDD), permitindo a avaliação formal da segurança de aplicação, os autores apontam que como vantagem da adoção desta metodologia está o equilíbrio entre os requisitos de segurança com os demais fatores, como a restrição de tempo para a disponibilização da aplicação no mercado e o orçamento do projeto (GEORG et al., 2010).

O artigo de Bradbury et al. (2020) apresenta uma arquitetura de referência (RA) que permite modelar aplicações da indústria aeroespacial e adota tal arquitetura para identificar a superfície de ataque do sistema. Assim, a análise da superfície de ataque é realizada usando árvores de ataque para especificar o caminho de ataque necessário para que um agente da ameaça alcance uma meta (BRADBURY et al., 2020).

3.2.1 Uso de DFD para análise de segurança

Apesar deste estudo ser centrado na aplicação do *Event Storming* como ferramenta de apoio à análise de segurança em microsserviços, é importante destacar que diversas abordagens utilizam Diagramas de Fluxo de Dados (DFD) como base para análise de superfície de ataque. Há uma evidenciada participação do DFD como modelo de representação arquitetural que facilita a identificação de vulnerabilidades associadas ao fluxo de dados em sistemas distribuídos.

Em *Reducing the Attack Surface for Private Data* (YEE, 2019), os autores utilizam o DFD para representar as interações entre processos, armazenamentos e fluxos de dados em sistemas que manipulam informações particulares. A análise proposta considera a contagem de componentes como forma de quantificar a superfície de ataque, por meio da fórmula N = m + n + k, onde m representa o número de fluxos de dados, n os armazenamentos e k os processos. A partir da visualização obtida com o DFD, estratégias arquiteturais são oferecidas para reduzir a exposição.

Por sua vez, o trabalho de Torkura et al. (2018) aplica o modelo STRIDE sobre DFDs e atores, fluxos e *trust boundaries*, permitindo identificar ameaças como falsificação de identidade, manipulação de dados e negação de serviço. Ainda neste contexto, o DFD serve como estrutura visual para o mapeamento de ativos, pontos de entrada e interações com componentes externos, apoiando o raciocínio sobre riscos e contramedidas.

Embora não utilize diretamente o DFD, o estudo *AndrAS: Automated Attack Surface Extraction for Android Applications* (TRAN; YSKOUT; JOOSEN, 2023) adota uma abordagem de representação gráfica interna baseada em componentes de sistemas *Android (Activities, Services, Receivers)*, funcionando como um DFD adaptado ao ambiente de aplicações móveis. Essa representação viabiliza a extração automatizada da superfície de ataque, reforçando a

importância de modelos visuais estruturados na avaliação de segurança.

Os quadros 3 e 4 a seguir relacionam os artigos dessa seção com esta dissertação.

3.3 Uso de IA no contexto de Engenharia de Software

O uso de *Large Language Models* (LLMs) como ferramenta de suporte na fase de requisitos traz ganhos de produtividade, principalmente nas etapas de documentação e ideação (RUIZ; PANADERO, 2024). No entanto, o risco de alucinação permanece presente, tornando a intervenção humana essencial no processo de engenharia de requisitos (BELZNER; GABOR; WIRSING, 2023).

Embora o uso de LLMs ofereça diversas vantagens, também apresenta certos riscos, como a dificuldade em lidar com detalhes contextuais e os potenciais vieses introduzidos pelo próprio algoritmo (ARORA; GRUNDY; ABDELRAZEK, 2024). A segurança dos dados é outra preocupação, especialmente ao usar LLMs de acesso público, pois informações sensíveis de processos e regras de negócio podem ser armazenadas em seus bancos de dados.

Apesar dos resultados promissores, o uso de LLM como ferramentas de suporte no processo de Engenharia de Software requer supervisão, curadoria e até mesmo revisão humana (BELZNER; GABOR; WIRSING, 2023). Desafios relacionados à responsabilidade e à ética também estão presentes ao empregar tais ferramentas em ambientes profissionais.

Quadro 3 – Trabalhos Relacionados — Análise de segurança nas etapas de design

Referência	**Contribuições**	**Limitações**	**Lacunas Exploradas na Dissertação**
Zhang (2023)	IAST com aprendizado de máquina para localizar/explorar vulnerabilidades em tempo real em microsserviços; políticas para reduzir falsos positivos e detectar anomalias.	Foco em execução/teste dinâmico (requer sistema implementado); não cobre elicitação colaborativa de requisitos nem modelagem visual no design.	Antecipar riscos ainda no design via Event Storming; diretrizes para incorporar preocupações de segurança antes da implementação.
Reselman (2022a), Reselman (2022b), Reselman (2022c)	Cinco princípios de design para microsserviços: responsabilidade única, limites discretos, transportabilidade, domínio/-dados próprios, efemeridade.	Série prática, não acadêmica; sem foco específico em segurança ou em oficinas colaborativas de requisitos; ausência de validação empírica.	Mapear princípios para heurísticas nas sessões de Event Storming; orientar decomposição e identificação de pontos críticos da superfície de ataque.
Rezapour et al. (2021)	Conceitos e requisitos de segurança em fog computing (revisão).	Escopo setorial (edge/-fog); pouco direcionamento para processos colaborativos de requisitos; distante de microsserviços genéricos.	Traduzir requisitos de segurança para checklists aplicáveis nas sessões de Event Storming em contextos de microsserviços.
Roman, Lopez e Mambo (2018)	Análise holística de ameaças em paradigmas de borda (fog, edge móvel, nuvem móvel).	Abordagem ampla e conceitual; limitada prescrição de atividades para elicitação/validação ágil.	Converter catálogos de ameaças em perguntas-guia e cartões de risco dentro da Big Picture do Event Storming.
Levcovitz, Terra e Valente (2016)	Técnica para identificar/-definir serviços e candidatos a microsserviços em sistemas monolíticos.	Foco em arquitetura/extração; não trata de segurança nem de dinâmica colaborativa de requisitos.	Usar a técnica para informar bounded contexts e artefatos do Event Storming, incluindo eventos/fluxos sensíveis à segurança.
Herrmann (2001)	Análise de fluxo de informações com UML para avaliar superfície de ataque em OO.	Alto formalismo; curva de aprendizado; menor aderência a práticas ágeis/colaborativas.	Oferecer alternativa leve (Event Storming) e ponte de saída: traçar elementos do ES para UML quando necessário.
Jürjens, Schreck e Bartmann (2008)	UMLsec: modelagem de requisitos de segurança e análise baseada em estereótipos/restrições.	Exige conhecimento especializado; baixa integração com oficinas de descoberta.	Usar resultados do ES para instanciar constructos UMLsec, mantendo colaboração no início e formalização posterior.
Georg et al. (2010)	AORDD (Aspect-Oriented Risk-Driven Development): avaliação formal equilibrando segurança, prazo e orçamento.	Complexidade de adoção; foco em desenvolvimento/gestão de risco, menos em descoberta colaborativa.	Inserir raciocínio risk-driven como passos nas diretrizes (critérios, trade-offs) durante o Event Storming.
Bradbury et al. (2020)	Arquitetura de referência (aeroespacial) e uso de árvores de ataque para identificar caminhos/superfície de ataque.	Escopo setorial; demanda detalhamento posterior (árvore de ataque) após decisões de design.	Usar ES para alimentar árvores de ataque com eventos/atores/limites; generalizar a prática além do domínio aeroespacial.

Fonte: Elaborado pelo autor.

Quadro 4 – Trabalhos Relacionados — Uso de DFD para análise de segurança

Referência	**Contribuições**	**Limitações**	**Lacunas Exploradas na Dissertação**
Yee (2019)	Uso de DFD para mapear processos/armazenamentos/fluxos em sistemas com dados privados; métrica para superfície de ataque; estratégias de redução.	Métrica pode simplificar realidades complexas; DFD é estático e menos colaborativo; foco principal em privacidade.	Oferece uma abordagem similar com mais aplicação de resultados, tanto para requisitos funcionais como não funcionais.
Torkura et al. (2018)	Aplicação de STRIDE sobre DFD (atores, fluxos, trust boundaries) para identificar ameaças (spoofing, tampering, DoS, etc.).	Depende de DFD detalhado; esforço manual; geralmente pós-design.	Pipeline ES → DFD → STRIDE: transformar a Big Picture em DFD e acoplar checagens sistemáticas nas diretrizes.

Fonte: Elaborado pelo autor.

Quadro 5 – Trabalhos Relacionados — Uso de IA/LLMs em Engenharia de Requisitos

Referência	**Contribuições**	**Limitações**	**Lacunas Exploradas na Dissertação**
Ruiz e Panadero (2024)	Apontam ganhos de produtividade no uso de LLMs nas etapas de documentação e ideação de requisitos.	O texto fornecido não discute riscos em profundidade ou mecanismos de mitigação; pressupõe necessidade de validação humana quando aplicado a requisitos.	Incorporação de diretrizes para aproveitar ganhos de produtividade com checkpoints de validação (human-in-the-loop), prompts orientados ao domínio e registro de decisões.
Belzner, Gabor e Wirsing (2023)	Evidenciam o risco de alucinação e a *necessidade de intervenção humana* (supervisão, curadoria e revisão) no processo de engenharia de requisitos; destacam aspectos de responsabilidade e ética.	Ênfase em alertas e salvaguardas, mas sem detalhar um processo operacional completo para oficinas de requisitos com LLMs.	Operacionalização via guidelines: papéis e responsabilidades, critérios de aceitação, revisão por pares e trilhas de auditoria para uso seguro e responsável de LLMs.
Arora, Grundy e Abdelrazek (2024)	Identificam riscos na gestão de detalhes contextuais e vieses dos modelos; alertam para a segurança de dados ao usar LLMs de acesso público.	Tratamento predominantemente de riscos; não apresenta, no trecho, um procedimento de mitigação integrado ao fluxo de elicitação colaborativa.	Recomendações de privacidade (não expor dados sensíveis), uso de instâncias/guardrails, e validação contextual ancorada na Big Picture do Event Storming.

Fonte: Elaborado pelo autor.

4

Metodologia

A pesquisa conduzida tem como objetivo propor guidelines para atender organizações que trabalham com microsserviços e métodos ágeis. O foco está na adoção de *Event Storming*, considerando uma arquitetura de microsserviços. Para a criação do artefato, que é a *guideline*, adotamos o *Design Science Research* (DSR) (PERRY; SIM; EASTERBROOK, 2006). Este método busca gerar e validar novos conhecimentos por meio da análise e implementação de artefatos práticos (HEVNER, 2007).

4.1 *Design Science Research*

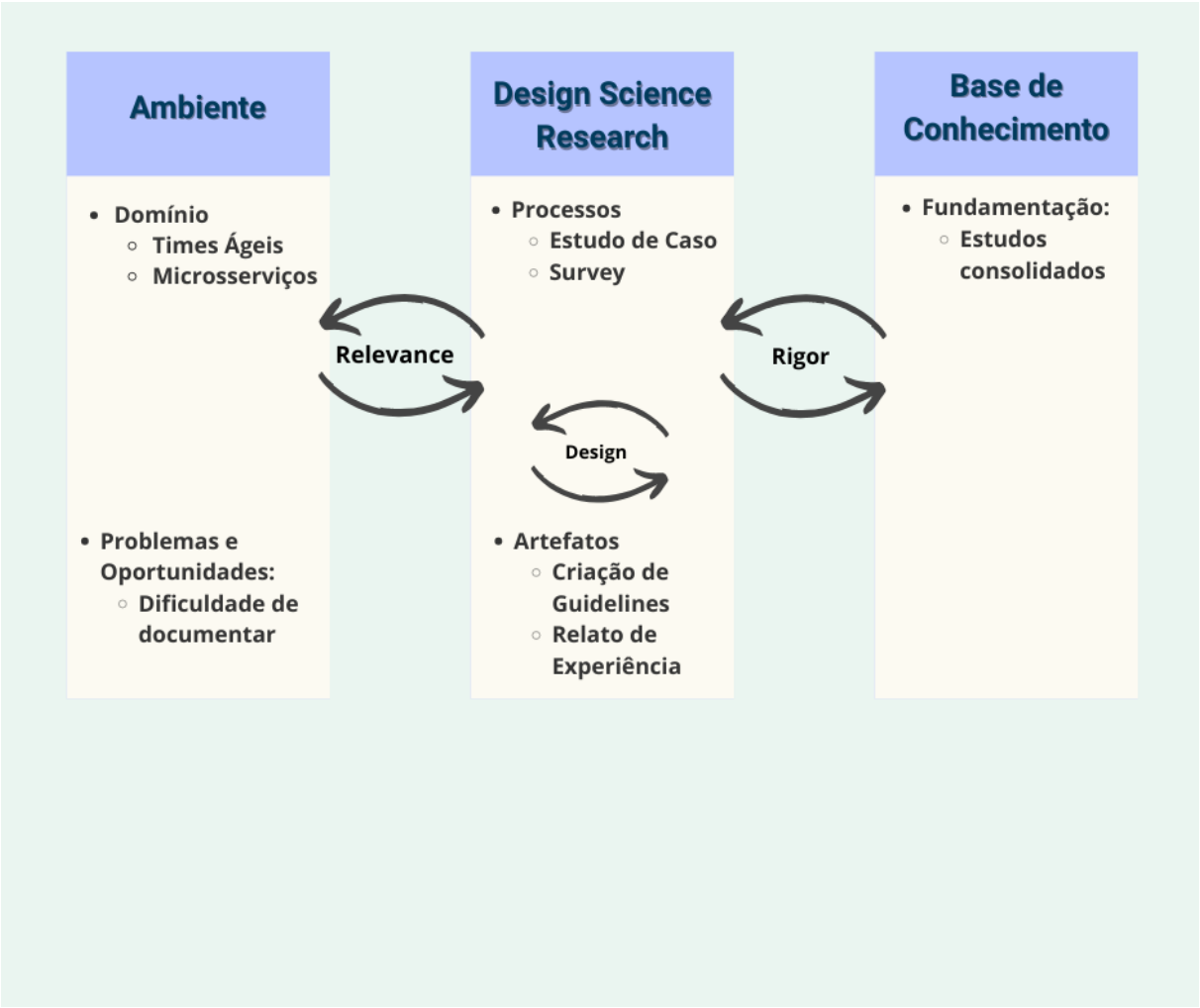
A partir de um contexto específico é possível determinar um problema de pesquisa que poder ser solucionado ou mitigado por meio da utilização de um artefato criado para este propósito. O design deste artefato é baseado em conhecimentos técnicos e científicos (PIMENTEL; FILIPPO; SANTOS, 2020) que serão levados em consideração no momento de validar o artefato como ferramenta para a solução ou a mitigação do problema em contexto.

A metodologia do *Design Science Research* evidencia dois momentos da pesquisa: Design e Avaliação. Para nosso design, a pesquisa será guiada pela importância e relevância do problema em questão (HEVNER, 2007). Quanto ao processo de avaliação, utilizaremos um método experimental e estudos de caso para colher os dados necessários para auferir a validade do artefato (HEVNER, 2007).

O artefato a ser desenvolvido neste estudo é as *Guidelines* para adoção de *Event Storming* para mapeamento de requisitos de microsserviços.

Os insumos para a construção deste artefato advêm dos resultados preliminares desta pesquisa, como fonte de conhecimento teórico e resultados técnicos e práticos. O esquema do desenvolvimento do DSR é apresentado na figura 6

Figura 6 – DSR Aplicado ao estudo



Fonte: Elaborado pelo autor.

Os estudos desta dissertação foram conduzidos conforme a Figura 6, iniciando com o estudo de caso para avaliar o *Event Storming* como ferramenta de apoio para a descoberta de requisitos não funcionais de desempenho e segurança. Em seguida, outro estudo foi conduzido coletando dados por meio de entrevistas com gestores de times ágeis que usam o *Event Storming* como ferramenta de requisitos funcionais.

Com os resultados destes estudos iniciais, somados à literatura encontrada sobre o *Event Storming*, a construção das *Guidelines* para utilização do *Event Storming* foi concebida como nosso artefato de DSR.

Como forma de avaliação do artefato, um novo estudo foi conduzido, desta vez um relato de experiência com *survey* envolvendo estudantes de graduação em Análise e Desenvolvimento de Sistemas. Estes estudantes foram treinados para executar o *Event Storming* e, em seguida, os resultados do *workshop* foram submetidos a um LLM a fim de gerar os requisitos funcionais, requisitos não funcionais e análises de qualidade.

Após a construção do artefato, uma nova etapa é esperada e trata-se da avaliação, validação e evolução deste artefato.

4.1.1 Estudos de Caso

Uma vez que o *Event Storming* é uma técnica recente com poucos resultados acadêmicos, somos encorajados a construir estes estudos de caso. Estudos de caso são indicados para entender um fenômeno de maneira mais aprofundada (PERRY; SIM; EASTERBROOK, 2006), sendo o presente estudo de caráter descritivo e exploratório (RUNESON; HÖST, 2009).

Dois estudos de caso foram conduzidos. O primeiro foi desenvolvido a partir de uma aplicação *open source* e serviu para se avaliar o desempenho do *event storming* como ferramenta de mapeamento de superfície de ataque.

O segundo estudo de caso foi conduzido uma entrevista onde foram entrevistados 3 responsáveis por times de desenvolvimento ágil que utilizam o *event storming* como ferramenta para delineamento de microsserviços, a fim de entender os desafios e particularidades da adoção do *Event Storming*.

Design do Estudo de Caso de Análise de Superfície de Ataque

Este estudo de caso utiliza como base um sistema de *e-commerce open source* ambientado em uma arquitetura de microsserviços. O objetivo desse estudo de caso é identificar a superfície de ataque deste software utilizando a abordagem do *Event Storming*, que, como mencionado anteriormente, foi projetada para definição de requisitos funcionais, porém o caráter exploratório deste estudo visa utilizá-lo para o mapeamento de requisitos não funcionais.

A seleção do software escolhido para este estudo de caso seguiu critérios alinhados à abordagem metodológica do *Design Science Research* (DSR), que entende a necessidade da construção e avaliação de artefatos em contextos reais ou simulados para validar sua aplicabilidade. Com esse objetivo, optou-se por utilizar uma aplicação *open source* baseada em arquitetura de microsserviços, com escopo funcional bem definido, componentes independentes e interface acessível para exploração técnica.

O sistema escolhido foi uma loja virtual com estrutura modularizada e documentação disponível, permitindo o mapeamento de eventos, comandos e atores a partir de uma análise técnica reversa. A escolha foi intencional por três razões principais:

- **Viabilidade técnica de análise exploratória:** a disponibilidade do código-fonte e da documentação de APIs permitiu a realização do *Event Storming* com engenharia reversa, mesmo sem a participação direta da figura de *stakeholders* do sistema.
- **Adequação ao objetivo do estudo:** o propósito era avaliar a aplicação do *Event Storming* na identificação de requisitos não funcionais de segurança, sendo que é de suma importância

que o sistema possuísse múltiplos pontos de entrada, fluxos de eventos distintos e arquitetura distribuída.

- **Caráter ilustrativo e replicável:** por se tratar de um sistema *open source*, outros pesquisadores podem reproduzir o estudo de caso, o que favorece a replicação da pesquisa.

Por meio da utilização do *Event Storming*, efetuamos o mapeamento de seus eventos. Estes eventos, como definidos por Brandolini (BRANDOLINI, 2018), são identificados e organizados preferencialmente em ordem cronológica. Em seguida, os comandos que irão disparar esses eventos são associados aos mesmos para que a partir daí os próprios eventos e as políticas (condições, decisões e gatilhos) entrem como ponte entre eventos distintos. Logo em seguida, atribuímos aos comandos os usuários que os consomem. Aqui é o primeiro ponto de atenção do estudo, visto que os usuários dos comandos sinalizam que aquele comando está potencialmente exposto.

Para registrar e conduzir a sessão de *Event Storming* foi utilizado o *software* Miro e os eventos e comandos foram mapeados a partir do código-fonte do sistema de *e-commerce*. As sessões foram assíncronas entre 3 desenvolvedores, o que justifica o uso do Miro uma vez que se trata de uma ferramenta online.

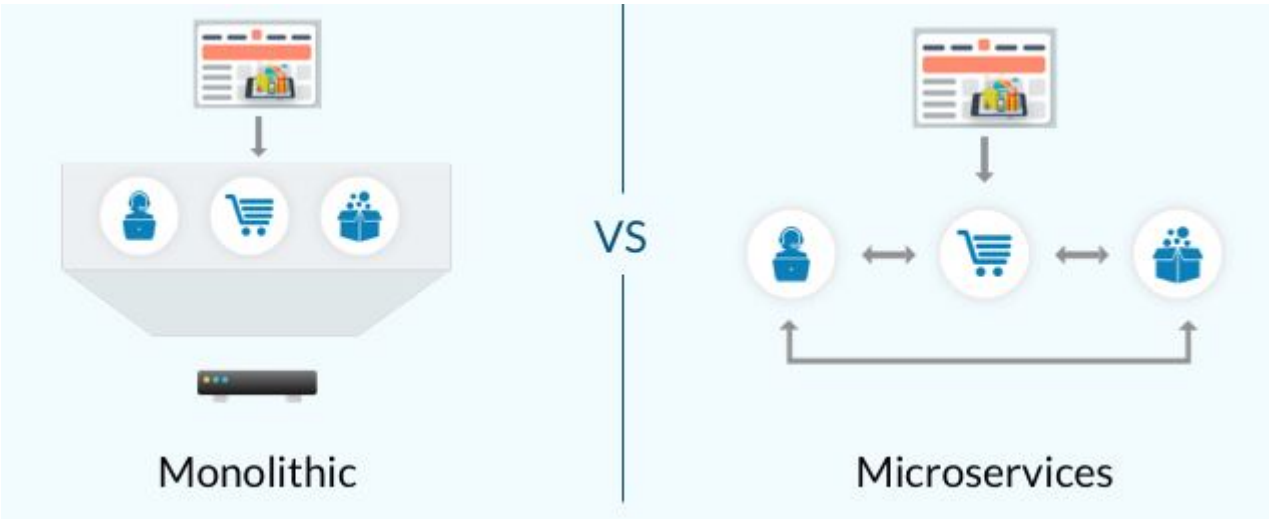
Nem todos os comandos serão iniciados diretamente por um usuário, e são esses comandos que utilizaremos para identificar a interface de acesso aos microsserviços estabelecidos. Dada a finalização desta etapa, quando todos os post-its conseguem formar a *Big Picture* do sistema, são atribuídos os contextos que irão resultar no fim das contas nos próprios microsserviços.

Estes comandos acessados por usuários, autenticados ou não autenticados, promovem o aumento da superfície de ataque naturalmente. Como o conceito do microsserviço prevê limites bem definidos e encapsulados (RESELMAN, 2022a), o caminho natural seria cada microsserviço cuidar da segurança em seu acesso. Mesmo um sistema de pequeno porte nesta arquitetura enfrentaria problemas em inserir em cada microsserviço sua própria camada de segurança.

A diversidade de serviços online, assim como a evolução da Internet, tornou a tecnologia uma parte natural na vida das pessoas, promovendo o desenvolvimento de aplicações web cada vez mais sofisticadas. Os fornecedores de serviços em nuvem, ao contrário da arquitetura monolítica, introduziram diversas tecnologias para atender a essas mudanças de requisitos com base na extensibilidade, escalabilidade e atributos modernos de qualidade de *software* (WEERASINGHE; PERERA, 2023).

Para usufruir da escalabilidade independente e da economia de recursos computacionais, os microserviços têm um papel crítico no roteamento de tráfego, sendo responsáveis por conectar sistemas através da comunicação do serviço de back-end real ou de dados.

Figura 7 – *Monolithic Vs Microservices*



Fonte: (DAFFODIL-SOFTWARE, 2020)

Design do Estudo de Caso com Entrevista

O principal objetivo deste trabalho preliminar é reunir mais informações sobre os impactos da adoção do *Event Storming* em times ágeis de desenvolvimento.

Para este estudo, avaliamos três representantes de equipes de desenvolvimento que atuam como facilitadores de reuniões de *Event Storming*. Esses representantes desempenham funções de gestão em equipes que trabalham remotamente. O Representante A trabalha em uma empresa no Brasil que desenvolve *software* próprio para o setor de aplicações financeiras. O Representante B atua em uma empresa brasileira que desenvolve *softwares* na área de segurança. O Representante C trabalha em uma *softhouse* localizada na Polônia. O critério utilizado para selecionar essas equipes para o estudo de caso foi a adoção da técnica de *Event Storming* durante a fase de requisitos. É importante salientar que este estudo de caso é qualitativo; exploramos a profundidade das informações de nossa amostra sem a necessidade de generalizar os resultados.

Preparamos um questionário com 12 perguntas subjetivas (ver Quadro 6), a fim de obter o máximo de informação possível sobre a adoção do *Event Storming* em seus ambientes de trabalho. O questionário foi elaborado de maneira similar aos questionários de criação de persona muito usados em técnicas de *design thinking* (SIRICHAROEN, 2020), porém o mesmo foi construído a fim de identificar algumas informações de forma direta e indireta. As questões 1 a 3 têm como objetivo entender como foi o início da utilização do *Event Storming*. A questão 4 visa descobrir quais ferramentas são utilizadas. As questões 5 e 6 visam entender quem são as pessoas que participam do *workshop*. As questões 7 e 8 nos ajudam a entender como o *Event Storming* se integra às práticas já existentes. A questão 9 avalia o grau de comprometimento com a atualização da *big picture*. A questão 10 visa explorar as novas possibilidades desenvolvidas a partir do *Event Storming*. A questão 11 verifica se o *Event Storming* é mensurável em algum aspecto, e por fim a questão 12 é focada em entender se a gestão está alinhada com a metodologia.

Quadro 6 – Guia da Entrevista

Questões da Entrevista	
Q	**Questão**
Q1	A adoção do *Event Storming* passou por algum estudo ou houve alguma outra motivação para sua escolha?
Q2	Quais os benefícios percebidos pela adoção do *Event Stroming*?
Q3	Quais os desafios encontrados quando adotaram o *Event Storming*?
Q4	Que ferramentas foram usadas,em caso de *home-office*, para conduzir o *Event Storming*?
Q5	Que tipos de pessoas foram envolvidas no *workshop* e por que?
Q6	A adesão do time ao *Event Storming* enfrentou alguma resistência?
Q7	As *User Stories* foram afetadas de alguma forma dada a aplicação do *Event Storming* em algum aspecto importante?
Q8	Sua equipe utiliza o *Event Storming* como ferramenta de suporte para alguma metodologia ou *framework* durante a fase de requisitos?
Q9	A *Big Picture* é atualizada ao longo do tempo? Qual é a frequência dessa atualização?
Q10	O *Event Storming* está sendo utilizado para outro propósito além da fase de requisitos?
Q11	Quais indicadores ou métricas foram obtidas com a adoção do *Event Storming*?
Q12	Como a gestão avalia o uso do *Event Storming*?

Fonte: Elaborado pelo autor.

4.1.2 *Guidelines*

O artefato a ser avaliado é na verdade o conjunto de *Guidelines* para adoção do *Event Storming* como ferramenta de descoberta de requisitos. Sua construção fundamenta-se na metodologia *Design Science Research* (DSR), que preconiza a construção de soluções práticas baseadas em evidências teóricas e empíricas. Assim, foram considerados como insumos para este artefato os resultados provenientes de três fontes: a revisão da literatura, um estudo exploratório de análise de superfície de ataque e entrevistas com gestores de projetos em contextos ágeis.

Partindo da visão metodológica, cada uma dessas etapas serve para identificar padrões, desafios e até mesmo lacunas que podem ser aproveitadas no artefato. A análise de superfície de ataque possibilitou compreender como é benéfico identificar os vetores de ataque ainda nas fases iniciais do design utilizando ferramentas de modelagem visual. Já as entrevistas contribuíram com uma visão prática sobre os óbices presentes na adoção de uma técnica relativamente nova.

Como uma contribuição específica, o artefato é criado de maneira a se alinhar com o contexto de times ágeis fornecendo um caminho estruturado e replicável para aplicação do *Event Storming*, onde aspectos importantes são trabalhados como: planejamento da sessão, identificação e organização dos elementos do domínio, uso estratégico de artefatos visuais e integração com ferramentas de suporte à documentação, no caso as LLMs.

4.1.3 Relato de Experiência com *Survey*

Este estudo adota uma abordagem exploratória de caráter qualitativo e quantitativo. realizada com estudantes do último período do curso de Análise e Desenvolvimento de Sistemas de uma instituição privada de ensino superior no estado de Sergipe. Para esta turma de alunos foi proposta como atividade de avaliação a construção de um software desde o planejamento, com foco em requisitos, até a entrega de uma versão estável no formato MVP (Minimum Viable Product).

A amostra foi composta por estudantes regularmente matriculados na disciplina de Tópicos Integradores, totalizando 17 alunos. Todos os participantes realizaram as atividades propostas em grupo, porém responderam individualmente ao questionário final de avaliação ao término da experiência.

Para execução das sessões de *Lean Inception* e *Event Storming*, os alunos usaram o programa Miro, inicialmente de maneira síncrona e presencial e posteriormente assíncrona e online. Os estudantes levaram cerca de 2 semanas para entregar os resultados das *Big Pictures*. O *survey*, respondido após a avaliação dos resultados obtidos, foi respondido individualmente através do *google forms*.

Quadro 7 – Questionário de avaliação individual

Questionário aplicado	
ID	**Questão**
Q1	É a primeira vez usando *Event Storming* como técnica de requisitos?
Q2	Já utilizou LLMs (ChatGPT, Gemini, Copilot etc.) para produzir requisitos de *software* anteriormente?
Q3	Como que você avalia sua capacidade de elaborar requisitos funcionais para o seu projeto de *software*?
Q4	Como que você avalia sua capacidade de elaborar requisitos não funcionais para o seu projeto de *software*?
Q5	De um modo geral como você avalia o resultado dos requisitos funcionais gerados a partir da LLM?
Q6	De um modo geral como você avalia o resultado dos requisitos não-funcionais gerados a partir da LLM?
Q7	As recomendações para garantir a qualidade do produto, foram adequadas para o projeto?
Q8	Voltaria a usar LLMs para gerar requisitos funcionais e não-funcionais?

Fonte: Elaborado pelo autor.

Utilizando as *guidelines* apresentadas no Quadro 11, a atividade proposta foi estruturada em três etapas principais desenvolvidas durante todo o período da disciplina de maneira ininterrupta e assistida pelo professor: capacitação e planejamento do projeto; geração automatizada dos requisitos através de LLM e; avaliação individual.

Os estudantes participaram de sessões introdutórias sobre *Lean Inception* (CAROLI,

2018), utilizadas para definição do escopo inicial do projeto e identificação do produto mínimo viável (MVP). Em seguida, estes mesmos alunos foram apresentados e treinados na aplicação da técnica *Event Storming*, com foco na construção da *Big Picture* do sistema a ser desenvolvido usando como base as funcionalidades identificadas na etapa de *Lean Inception*.

Com a *Big Picture* construída, os estudantes utilizaram um LLM configurado e treinado com instruções específicas sobre *Event Storming* e engenharia de requisitos. A interação ocorreu com um *prompt* estruturado com o resumo do projeto e as *big pictures* correspondentes. A partir disso, foram gerados requisitos funcionais, não funcionais e recomendações sobre qualidade do produto.

Após a obtenção dos resultados da LLM, cada aluno avaliou a qualidade dos requisitos gerados e respondeu a um questionário individual contendo itens fechados em escala *Likert* (nível de adequação, percepção de dificuldade, intenção de reutilização). As questões versam sobre a experiência de uso da técnica *Event Storming* e autoavaliações quanto à capacidade de elaboração de requisitos com e sem auxílio da IA. As questões apresentadas encontram-se no Quadro 7.

Os dados coletados foram analisados principalmente por meio de estatística descritiva. Em seguida, a análise cruzada entre algumas variáveis e uma análise qualitativa mais simples das percepções relatadas, categorizando aspectos como utilidade, limitações, autonomia crítica e intenção de uso futuro.

5
Resultados

Este capítulo apresenta os principais resultados da pesquisa conforme as etapas definidas pela metodologia apresentada, o *Design Science Research*. Inicialmente são apresentados os resultados dos estudos preliminares que contribuíram para ampliar a visão acerca dos desafios práticos do uso do *Event Storming*. Em seguida, o processo de construção das *guidelines*, o artefato em questão e o principal resultado desta pesquisa. Para a construção desse artefato foram utilizados tanto os dados dos estudos até então desenvolvidos como também a literatura existente. Por fim, é apresentado o relato de experiência com fins de avaliação do artefato utilizando LLM como ferramenta de apoio.

5.1 Aplicando o *Event Storming* na analise de superficie de ataques

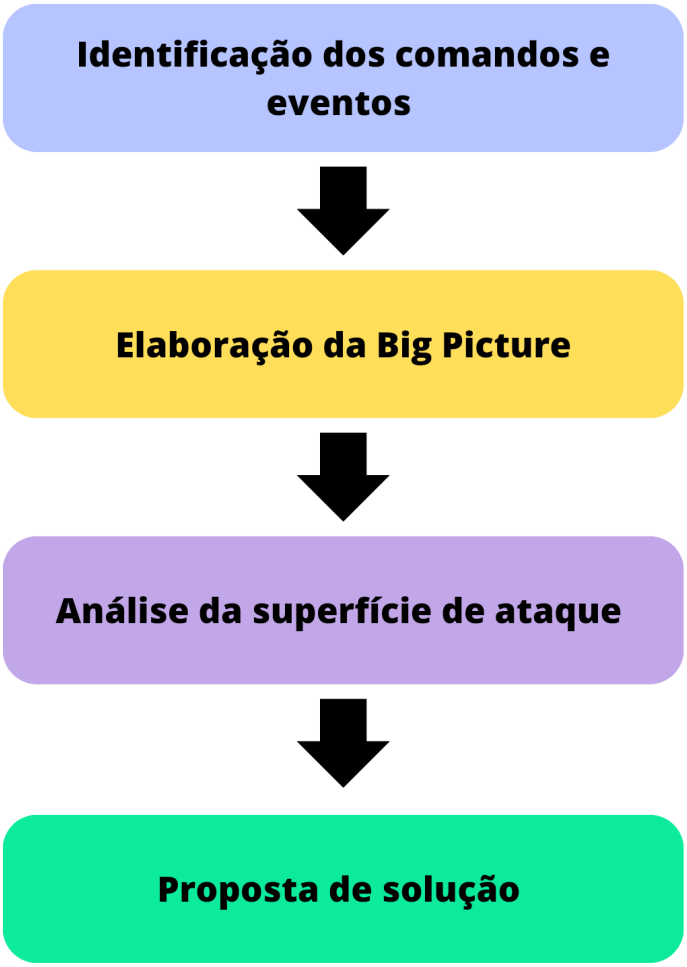
A utilização do *Event Storming* não precisa se limitar ao contexto da Engenharia de Requisitos. No caso apresentado, o *Event Storming* é utilizado a partir de um sistema pronto, ou seja, de uma loja *online* numa espécie de engenharia reversa para que seja possível identificar os *endpoints* que estão expostos na intenção de mapear a superfície de ataque.

Na intenção de analisar a forma como o *Event Storming* pode ser usado para identificar requisitos não funcionais, um estudo de caso foi conduzido (RUNESON; HÖST, 2009) em caráter exploratório e descritivo. O formato da metodologia do estudo de caso é ilustrado na Figura 8

5.1.1 Avaliação

Conforme abordado nesta seção, o uso de microsserviços no formato analisado expõe a aplicação ao permitir que os usuários acessem os serviços de forma direta. Para cada serviço exposto, uma nova abordagem de segurança deve ser implementada, uma vez que os microsserviços

Figura 8 – Metodologia Utilizada



Fonte: Elaborado pelo autor.

são responsáveis pelos seus próprios dados (RESELMAN, 2022a) e estabelecem limites bem definidos. Como cada microsserviço deve cuidar de uma única responsabilidade dentro do sistema, seria necessário mais um microsserviço para cada exposição, menos que este princípio de responsabilidade única seja ignorado. Então se há a previsão do deslocamento da segurança do microsserviço, nada impede em si que essa camada de segurança se comprometa com toda a segurança dos microsserviços expostos. Essa solução se encaixa no padrão *API Gateway* (OZKAYA, 2022) uma vez que favorece uma camada de proteção centralizada, permitindo novamente que os microsserviços que foram expostos se preocupem apenas com a sua única responsabilidade.

Por intermédio deste padrão (figura 9), a segurança desses microsserviços é movida para um canal de mapeamento de tráfego externo que se encarregará de rotear as requisições para tráfego interno com um ponto único de entrada para um grupo definido de microsserviços/serviços. Ou seja, o *front-end* (usuário) irá se comunicar com serviços distintos, oportunidade em que terá que armazenar todas as URLs dos serviços de *back-end* e assim permitir conhecer e gerenciar os

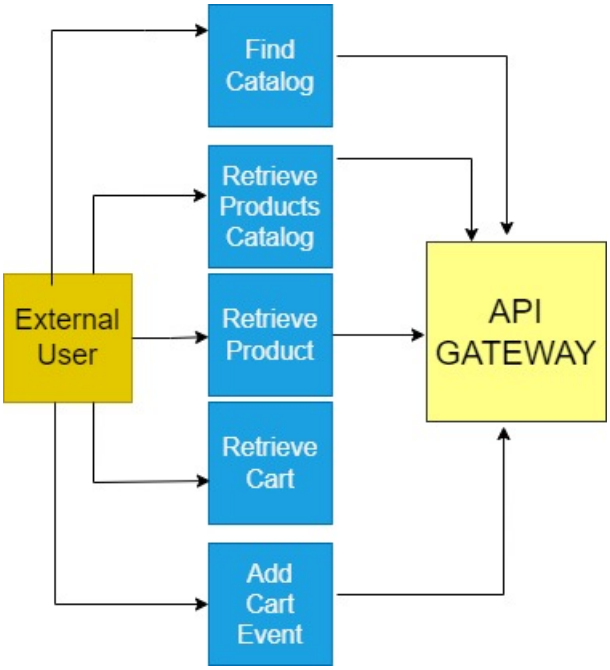
padrões de rotas. A complexidade dos microsserviços e, consequentemente, de todo o sistema diminui em razão dessa centralização das ações de segurança (DAWEI et al., 2021).

Para centralizar as requisições e a forma como o *front-end* interage com o *back-end*, o padrão *API Gateway* (OZKAYA, 2022) fornece um *proxy* reverso para redirecionar ou rotear solicitações para seus *endpoints* de microsserviços internos, mapeando internamente as solicitações e coletando as solicitações em um único ponto de entrada, encaminhando para microsserviços internos.

Esta solução representa a importância dos serviços primordiais como autenticação, autorização, políticas de segurança, balanceamento de carga, gerenciamento de cache, resolução de dependências, *rate limiting e throttling, log, trace,* correlação, transformações de *Headers*, *query string, IP Whitelist* e SSL/TLS externo, *clear text* interno. Assim, a arquitetura de microsserviços facilita a comunicação, em que um único ponto de entrada permite o controle, ainda que de forma transversal, possibilitando que tudo o que for implementado pela API seja reduzido ao acoplamento de vários microsserviços, cujos benefícios são: Abstração de protocolos; Baixo acoplamento a serviços internos e; Redução de *Round-trip time* (RTT).

A união entre microsserviços e a *API Gateway* fornece uma camada adicional de proteção (DAWEI et al., 2021) contra vetores de ataque, injeção de SQL, explorações do Analisador de XML e ataques de negação de serviço (DDoS).

Figura 9 – Sugestão de solução implementando o padrão *API Gateway*



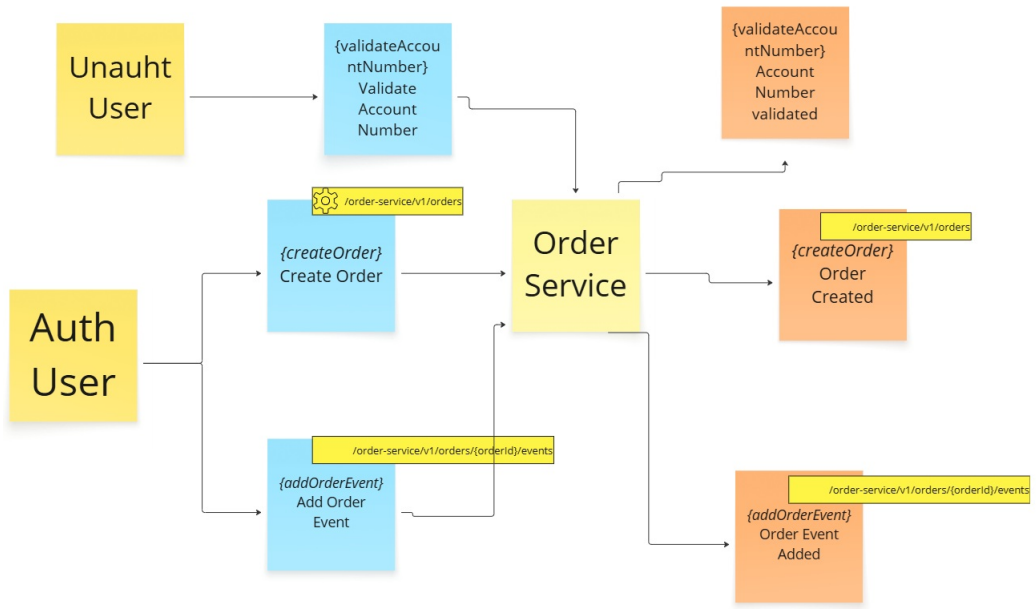
Fonte: Elaborado pelo autor.

5.1.2 Análise dos Resultados

A aplicação da técnica de *Event Storming* no mapeamento da superfície de ataque oferece inúmeras vantagens que contribuem para a segurança de sistemas desde a fase inicial de definição de requisitos, muito antes do efetivo início do desenvolvimento. Neste contexto, é relevante destacar o uso dos Diagramas de Fluxo de Dados (DFD) como ferramentas para a identificação da superfície de ataque, como discutido em estudos anteriores (THEISEN et al., 2017; LIU et al., 2020).

No entanto, o *Event Storming* se diferencia por ser um modelo que enfatiza a comunicação entre os *stakeholders*, permitindo que todos os envolvidos analisem coletivamente o sistema. Isso possibilita a identificação ágil e iterativa dos potenciais pontos de vulnerabilidade do sistema, figura 10. Assim, com a *big picture* feita de forma colaborativa, foi possível examinar os comandos, eventos e seus usuários, permitindo, assim, identificar um total de 17 ações, das quais 13 estão expostas na Internet por meio do acesso à API. Demonstrando que a *big picture* pode ser um artefato valioso na análise de superfície de ataque.

Figura 10 – Primeiro Cenário do mapeamento do código-fonte do *e-commerce*



Fonte: Elaborado pelo autor.

Ao contrário da abordagem representada pelo *Interactive Application Security Testing* (IAST) (ZHANG, 2023), que requer a existência de um sistema implementado, a análise da superfície de ataques, realizada na fase de design de software, representa uma prática que ostenta uma série de méritos para a segurança e eficácia de sistemas. Esta prática permite a identificação dos elementos suscetíveis a ataques antes de sua efetiva implementação, configurando, assim, uma abordagem proativa na mitigação de vulnerabilidades e riscos de segurança desde os estágios

iniciais do desenvolvimento. Já o trabalho de Bradbury et al. (2020) traz uma abordagem que usa um modelo de referência, apesar de eficiente, tal modelo está aplicado em um contexto específico. Em geral, os trabalhos adotam UML (HERRMANN, 2001; JÜRJENS; SCHRECK; BARTMANN, 2008), criando um formalismo para análise de superfície de ataques. Como diferencial, este trabalho adotou o *Event Storming*, que traz um formalismo menor que a UML, porém, permite a análise de superfície de ataques de forma integrada aos modelos ágeis.

Vale ressaltar que, embora seja amplamente aconselhável a aplicação do *Event Storming* antes da etapa de implementação, este modelo também pode ser empregado como uma ferramenta de reengenharia, conforme ilustrado no estudo de caso em questão. Neste cenário, o método é aplicado a um produto já existente, no qual se realiza um *workshop* para identificar os atores, comandos, eventos, políticas e outros recursos envolvidos. Posteriormente, a criação de uma *"big picture"* da visão atual do sistema viabiliza a análise da superfície de ataques. Como resultado, com base nas conclusões dessa análise, uma nova *"big picture"* é gerada, contendo recomendações para a refatoração do sistema.

Este modelo, embora envolva um esforço adicional, proporciona aos *stakeholders* a oportunidade ágil de avaliar o estado atual do sistema. Isso permite a tomada de decisões embasadas no impacto das exposições identificadas na superfície de ataque e a seleção das alterações a serem efetuadas, promovendo, assim, uma abordagem fundamentada na gestão proativa da segurança do *software*.

5.2 Estudo de Caso com Entrevista: Requisitos com Event Storming em Times Ágeis

Um fator adicional a ser considerado em relação à UML é que os benefícios máximos de sua implementação estão diretamente ligados à sua formalidade (GARDNER; BLACKWELL; CHURCH, 2020). Embora essa abordagem inicialmente ofereça maior consistência e precisão, a atualização dos artefatos da UML nem sempre é realizada de maneira adequada, o que resulta em documentação imprecisa e desatualizada.

À medida que as práticas de desenvolvimento de software evoluem em direção a abordagens ágeis, as demandas por modelagem de software também sofrem transformações. Nesse contexto, emergem novas formas e metodologias para atender às necessidades de compreensão rápida e precisa dos requisitos e demandas de software. Destaca-se, especialmente, o *Domain-Driven Design*, que oferece *workshops* objetivos, culminando na produção de um artefato final denominado *Big Picture* (Figura 11), facilmente compreendido devido à sua construção em linguagem ubíqua (HOFER; SCHWENTNER, 2021). Através dessa linguagem ubíqua, os *stakeholders* são incluídos de forma ativa no processo de levantamento de requisitos.

Embora o *Event Storming* tenha uma formatação esperada (BRANDOLINI, 2018), sua aplicação não é rígida, tornando-o altamente adaptável às necessidades das equipes de desenvolvimento. Decidir os pontos onde esta técnica pode ser adaptada torna-se um desafio para o facilitador do *workshop*.

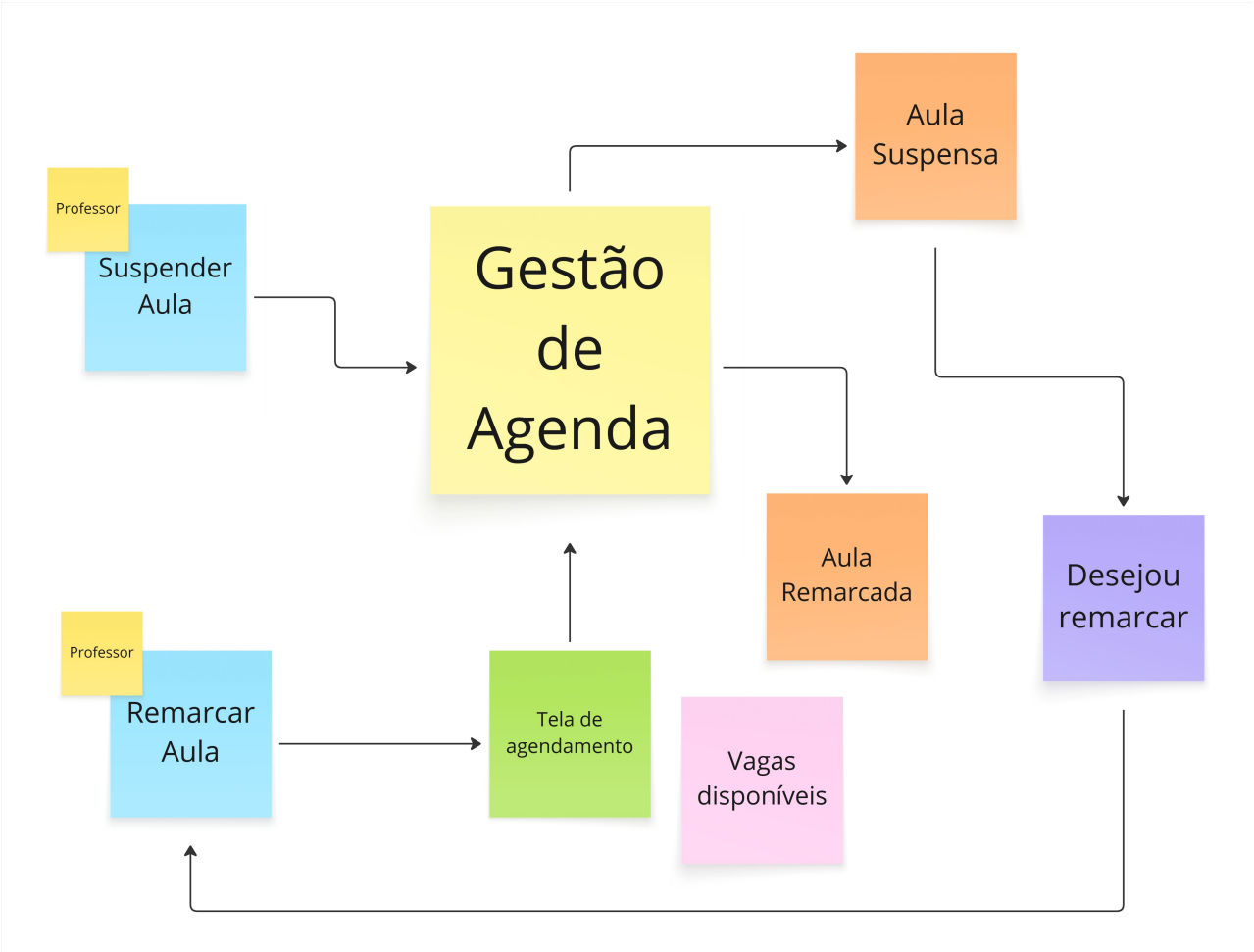
Para este estudo de caso, procuramos reunir as experiências encontradas na adoção do *Event Storming* por meio da análise dos procedimentos adotados por algumas equipes de desenvolvimento de software que utilizam o *Event Storming* como técnica de análise de requisitos. Para organizar nosso estudo de caso, estabelecemos 3 questões de pesquisa apresentadas no Quadro 8.

Questões de Pesquisa	
NR	**Questão**
QP1	Quais os benefícios mais evidentes percebidos após a adoção do *Event Storming*?
QP2	Quais os desafios iniciais ao adotar o *Event Storming*?
QP3	Existe algum outro uso do *Event storming* para algo além do estabelecimento de requisitos funcionais?

Quadro 8 – Questões de Pesquisa

Essas coletas servem principalmente como guia para aprimorar estudos do *Event Storming* e para escolher o formato do *workshop* a fim de obter os melhores resultados. O processo de coleta está descrito na figura 12. Serão analisados os pontos positivos e negativos enfrentados pelos entrevistados quanto à adoção do *Event Storming* como ferramenta utilizada na fase de Requisitos.

Figura 11 – Exemplo de *Big Picture* do *Event Storming*



Fonte: Elaborado pelo autor.

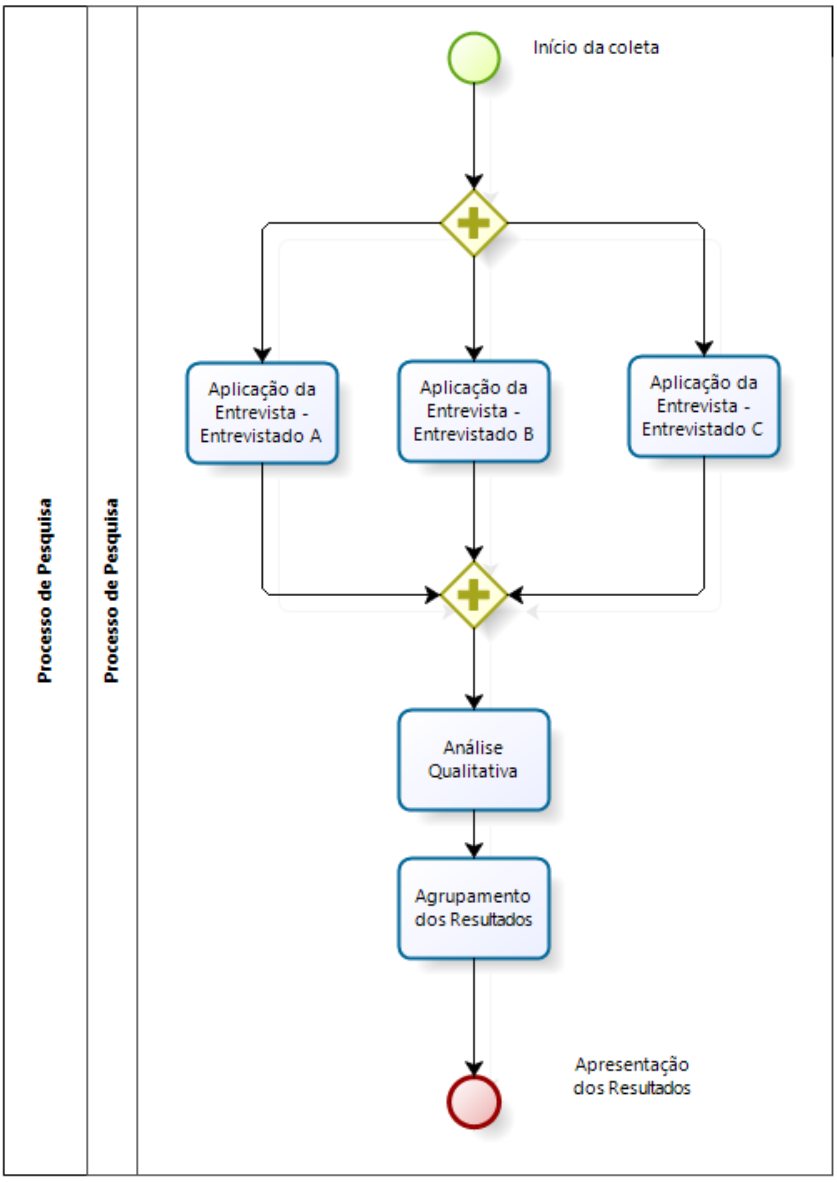
5.2.1 Análise Qualitativa

De acordo com as respostas do questionário, é possível identificar alguns benefícios do emprego do *Event Storming*. A seguir, analisamos o resultado de cada pergunta do questionário.

Começamos o questionário com uma simples pergunta sobre a adoção do *Event Storming*. Por se tratar de uma abordagem nova, é muito importante saber qual foi a motivação para sua escolha diante de várias outras técnicas.

O facilitador A revela que já vem observando a adoção do *Event Storming* em outras empresas e que este fator de observação foi importante para a escolha do *Event Storming*. Quanto aos facilitadores das equipes B e C, uma fase de estudo e comparação com outras abordagens culminou na escolha do *Event Storming* por ser uma técnica de fácil explicação, podendo ser explicada durante o processo. A equipe B valoriza a escolha do *Event Storming* por conta da aderência aos modelos ágeis de metodologia. A equipe C, que recebeu recomendações de outros profissionais para a adoção do *Event Storming*, considera como fator de escolha a possibilidade de incluir os *stakeholders* em todo o processo e a evidente melhoria do detalhamento das histórias

Figura 12 – Processo de Pesquisa



Fonte: Elaborado pelo autor.

de usuários.

Como um benefício, conforme mencionado pelo facilitador A, o *Event Storming* pode ajudar a identificar alguns problemas, tais como o trabalho duplicado em grandes equipes e limites de responsabilidade de microsserviços. As histórias de usuários foram melhoradas porque o *Event Storming* incentiva o *Domain-Driven Design* e o *Domain Driven Storytelling*. Combinado com a *Big Picture*, as histórias de usuários tornam-se ilustradas, tornando-as mais fáceis de entender. Uma dessas melhorias é sobre a possibilidade de dividir uma história em atividades mais amigáveis para os desenvolvedores, que podem ser separadas de acordo com o seu tipo, algo como *backend*, *frontend*, outro serviço, etc.

Todos os facilitadores admitem que reunir membros de mais de uma equipe já é um trabalho árduo, e adicionar o usuário final pode ser ainda mais difícil, mas esta é a melhor configuração de reunião que pode ser feita. A proposta original do *Event Storming* é reunir pessoas em uma sala com um grande quadro, vários blocos de post-its, marcadores de quadro e uma mesa com lanches para um simples *coffee-break*. Este é um cenário idealizado quando os participantes têm um trabalho presencial com uma certa disponibilidade de tempo. É possível imaginar como unir diferentes tipos de pessoas pode culminar em dificuldades de agenda, que podem ser um obstáculo quando pensamos em marcar uma reunião com pessoas da área de negócios, equipe técnica e usuários do aplicativo. A possibilidade de preparar a reunião do *Event Storming online* pode mitigar essa dificuldade, sendo então necessário o uso de ferramentas que permitam a coleta de informações e a construção da *Big Picture*.

Os membros da equipe entrevistados utilizam algumas ferramentas para auxiliar o evento *online*, como Figma, DrawIO e Miro, que são usadas para construir diagramas. A documentação textual, como notas de domínio, pode ser produzida em Markdown.

Quadro 9 – Ferramentas usadas no *Event Storming*

Ferramentas Usadas		
Ferramenta	**Uso**	**Link**
Miro	*Big Picture*	https://miro.com/
Draw IO	*Big Picture*	https://www.drawio.com/
Markdown	Documentação	https://www.markdownguide.org
Figma	*Big Picture*	https://www.figma.com/
Jira	Gestão Ágil	https://www.atlassian.com/br/jira
Microsoft Teams	Videoconferência	https://teams.microsoft.com/v2/

Fonte: Elaborado pelo autor.

Dentro das equipes dos entrevistados, as reuniões de *Event Storming* são conduzidas com uma variedade de pessoas que representam diversos papéis. Obviamente há pessoas da equipe de desenvolvimento, mas também há espaço para as equipes de negócio, de segurança e qualidade. O facilitador C amplia ainda mais essa participação envolvendo os clientes e *stakeholders* externos. Para ele, o *Event Storming* é onde todas as pessoas passam a conhecer todas as histórias.

Quanto à adesão do *Event Storming* por parte das equipes, todos os facilitadores entrevistados relatam a resistência inicial por parte de pessoas mais antigas que já estão acostumadas com métodos de trabalho já estabelecidos. O facilitador C teve a experiência de conduzir as primeiras sessões com times com pouca participação, por conta da timidez e não da resistência. Para o facilitador A, foi necessário um esforço que envolveu treinamentos, indicadores de sucesso e até mesmo total resistência. Para o facilitador B, a resistência foi quanto à constante atualização dos artefatos do *Event Storm*: havia receio quanto à alteração da *Big Picture*.

Um ponto positivo amplamente observado por todos foi a melhora nas *User Stories* como um todo, desde o detalhamento dessas *User Stories*, passando pela decomposição em tarefas menores e mais focadas, assim como na reescrita dessas *user stories*. Outro benefício importante para estas *User Stories* é a possibilidade de se perceber o impacto que cada uma delas pode ter no produto como um todo.

A associação do *Event Storming* com outras metodologias ou *frameworks* tem sido possível e o principal motivo está em sua simplicidade, tornando-o facilmente encaixável em processos do BDD, SCRUM e atividades do *Lean Inception* (CAROLI, 2022). Até mesmo como suporte para diagramas da UML quando se deseja trabalhar com algoritmos mais complexos.

Ainda de acordo com os relatos dos facilitadores, a *Big Picture* é atualizada frequentemente, em períodos que variam de 1 a 6 meses, e seu uso é estendido para outras análises, como área de ataque de superfície, gargalos de desempenho e dimensionamento de equipe. No entanto, uma das equipes necessita evitar atualizações da *Big Picture* porque, como uma *soft-house*, mudanças no grande quadro representam um novo custo.

Para além do uso na análise de requisitos, cada entrevistado proporcionou outra utilidade para o *Event Storming*. Para o facilitador A, o *Event Storming* permite que sejam feitos estudos de *Event Driven Architecture* e *Event Source Patterns*. Para o entrevistado B, os resultados do *Event Storming* proporcionam uma série de desdobramentos que vão desde a Análise de Superfície de Ataque, indicadores de performance e automação para testes. O facilitador C promove melhores estimativas de prazos para as atividades geradas no *Event Storming*.

As gestões das equipes dos facilitadores B e C estão satisfeitas com os resultados apresentados e na forma como a comunicação com os desenvolvedores é facilitada. Porém, para a equipe do facilitador A, pouco valor é percebido, sendo assim necessário algum trabalho por parte da equipe para favorecer esse entendimento.

5.2.2 Avaliação dos resultados

Os resultados analisados, condensados no Quadro 10, serviram para responder às questões de pesquisa apresentadas no Quadro 8. Como a metodologia deste estudo é de caráter exploratório e nosso universo de entrevistados é limitado, não é possível caracterizar ou generalizar padrões. Entretanto, é a partir dos resultados destas questões de pesquisa que podemos embasar novas

pesquisas de caráter descritivo.

Relação de resultados resumidos	
QP1 - Quais os benefícios mais evidentes percebidos após a adoção do Event Storming	
Adaptação e Flexibilidade	O *Event Storming* possui regras simples para condução do *workshop* e para a representação da *big picture*
Comunicação Aprimorada	A metodologia favorece a comunicação entre equipes distintas
Historias de Usuários Aprimoradas	Histórias de Usuários podem receber detalhes técnicos e serem subdivididas em tarefas menores
Sinergia com outras metodologias	O ES se mostrou compatível com outras metodologias como o *Lean Inception*
Engajamento de time	Participação de pessoas de diferentes áreas, incluindo *stakeholders*
QP2 - Quais os desafios iniciais ao adotar o Event Storming?	
Provas de Eficácia	Os participantes precisam ser convencidos de que o *Event Storming* traz bons resultados.
Agenda de Reuniões	Conforme o grupo selecionado se torna maior e mais heterogêneo os conflitos de agenda podem aparecer
Atualizações de Artefatos	Uma baixa participação pode resultar em uma *Big Picture* inadequada ou obsoleta
QP3 - Existe algum outro uso do Event Storming para além do estabelescimento de requisitos funcionais?	
Identificação de Gargalos de Desempenho	Ao observar a *Big Picture* é possível identificar comandos que recebem muitas requisições como possíveis pontos de gargalo de desempenho
Identificaçao de Área de Superfície de Ataque	Comandos que são disparados por usuários podem ser interpretados como *endpoints* expostos
Dimensionamento de Equipe	Conforme as histórias de usuários recebem detalhamento técnico com o *Event Storming*, um dos frutos deste detalhamento é a capacidade de estimar o esforço necessário para conclusão de suas tarefas

Quadro 10 – Resultados da Entrevista

QP1 - Quais os benefícios mais evidentes percebidos após a adoção do Event Storming

Adaptação e Flexibilidade: O *Event Storming* possui um conjunto de regras simples quando comparado a outras metodologias similares. Principalmente quando comparamos com os diagramas UML que requerem um tempo maior de aprendizado para sua interpretação correta. Ter essa linguagem de representação facilitada possibilita que o próprio método do *Event Storming* seja expandido e adaptado conforme as necessidades das equipes.

Comunicação Aprimorada: Um dos temas abordados durante as entrevistas foi sobre

como a comunicação entre as equipes é facilitada com o *Event Storming*. Unir diferentes papéis de uma equipe e até mesmo os clientes que não conhecem a fundo detalhes técnicos, o *Event Storming* se apresenta como uma viável alternativa para a comunicação, permitindo que a *Big Picture* seja facilmente interpretada.

Ou seja, a *Big Picture* é além de uma documentação, uma poderosa ferramenta de comunicação. Sendo assim, ela precisa de atualizações sempre que necessário.

Histórias de Usuário Aprimoradas: Existe uma preocupação com a ambiguidade das histórias de usuários, isto é, as histórias podem ocultar detalhes técnicos importantes (AMNA; POELS, 2022). As histórias de usuários podem ser divididas em unidades menores, gerenciadas por diferentes equipes, por isso é necessário detalhar essas tarefas e associar cada tarefa à sua respectiva equipe.

Com a *Big Picture*, é muito fácil descobrir e associar todas essas tarefas corretamente. Uma história de usuário pode ser dividida em um número maior de tarefas diferentes; cada tarefa pode ser representada como um post-it de comando (os azuis) e as equipes podem verificar se suas tarefas fazem parte de um contexto de dependência.

Uso em conjunto com outras metodologias: A flexibilidade e simplicidade do *Event Storming* o permitem ser usado em conjunto com outras metodologias ou *frameworks*. Como é possível detalhar as *User Stories* a um nível técnico, tanto nas metodologias do SCRUM como na metodologia do *Lean Inception* existe um espaço para o *Event Storming* (CAROLI, 2022).

Engajamento do time: O *Event Storming* permite a participação ativa de diversos tipos de pessoas, desde a equipe de qualidade e segurança até mesmo os *stakeholders*. Porém, conseguir essas participações envolve diversos esforços e isso precisa ser levado em consideração conforme os eventos demandem a participação de mais pessoas. Felizmente, a participação de diferentes pessoas foi sinalizada como uma experiência muito positiva, devendo sempre ser estimulada pelos facilitadores.

QP2 - Quais os desafios iniciais ao adotar o Event Storming?

Desafios preliminares: Conforme mencionado no tópico anterior, a inclusão de pessoas dos mais distintos papéis é encorajada no *Event Storming*, tornando mais amplo o processo de levantamento de requisitos. Como envolvemos mais pessoas além da equipe de desenvolvimento, as equipes de negócios, segurança, qualidade e até mesmo clientes e *stakeholders* externos, a dinâmica do evento presta um grande serviço de valor com a inclusão de seus pontos de vista únicos.

Porém, para atingir este patamar de altas contribuições com grandes probabilidades de satisfação, um esforço considerável é necessário no sentido de convencer essas pessoas de que a participação delas é importante e bem-vinda, mesmo quando se trata de profissionais fora da área de tecnologia.

Provas de Eficácia: Outra dificuldade encontrada é quanto à eficácia da técnica. Muitos profissionais têm resistência a conhecer técnicas novas de levantamento de requisitos e, quando se apresenta o *Event Storming* como uma ferramenta eficiente, faz-se necessário apresentar provas de valor, treinamentos e outros incentivos à participação.

E por fim, o convencimento do público da gestão se torna uma tarefa a mais, visto que para se reunir pessoas em uma sessão de *Event Storming* é importante reforçar a possibilidade dos ganhos que essa abordagem proporciona.

Como são desafios que se apresentam no primeiro contato com o *Event Storming*, é muito importante iniciar um trabalho de conscientização antes, com palestras, treinamentos e até mesmo resultados de pequenos projetos.

Reuniões do Event Storming: A disponibilidade de tempo das pessoas envolvidas é uma preocupação recorrente. As reuniões online tendem a ter mais participação do que as reuniões presenciais, embora estas reuniões, quando presenciais, incentivem o foco e a participação.

Ainda sobre reuniões online, também são indicadas para grupos numerosos desde que amparadas com algum software para registro dos cartões do *Event Storming*, como o Figma ou o Miro. Apesar de ser um momento muito propício para a inclusão de diversas pessoas para contribuir com o *workshop*, há de ser avaliado o custo de alocar o tempo dessas pessoas.

Atualizações: A adesão inicial ao *Event Storming* pode apresentar desafios, especialmente em termos de participação. Isso pode ser devido a uma variedade de fatores, incluindo a resistência à mudança, a falta de familiaridade com a técnica ou a percepção de que ela é desnecessariamente complexa ou demorada.

Independentemente do fator que ocasiona essa falta de participação, o baixo engajamento evidencia a necessidade de atualização da *Big Picture*, uma vez que uma das vantagens da técnica é a fácil atualização de seus artefatos.

QP3 - Existe algum outro uso do event storming para algo além do estabelecimento de requisitos funcionais?

Identificação de gargalos de desempenho: É possível por meio de uma simples observação da *Big Picture*, quais são os comandos e eventos que estão concentrando as requisições de fluxo. Potenciais gargalos de desempenho ficam evidentes quando um único comando ou evento concentra uma quantidade numerosa de requisições de outros comandos dentro e fora do *Bounded Context*.

É importante notar que estes gargalos de desempenho podem ser identificados tanto nos casos onde o *Event Storming* é usado como ferramenta de descoberta de requisitos quanto na engenharia reversa para investigação das causas das anomalias de desempenho.

Análise de superfície de ataque: Os comandos representados na *Big Picture* podem ser iniciados como resposta a eventos e políticas de forma segura. Por outro lado, os comandos

podem ser iniciados por um usuário externo como um ator da UML. Estes comandos iniciados por usuários podem ser catalogados pela equipe de segurança que deve levá-los em consideração por aumentar a área de superfície de ataque.

Dimensionamento de Equipe: Como um benefício do aprimoramento das *User Stories*, a decomposição das histórias em comandos e eventos do *Event Storming* nos permite visualizar o quanto de esforço será necessário para concluir esta história. Essa prática permite tanto definir parâmetros como grau de esforço e complexidade (CAROLI, 2022), como também contribui para o dimensionamento de equipes.

5.3 *Guidelines* para adoção do *Event Storming*

Com base nos estudos de Análise de Superfície de Ataque presente na seção 5.1, e no Estudo de Caso de Requisito com *Event Storming* em Times Ágeis presente na seção 5.2, com a intenção de reproduzir os benefícios e mitigar, ainda que parcialmente, algumas preocupações recorrentes, uma lista de *guidelines* foi produzida e apresentada no Quadro 11.

***Guidelines* Propostas**		
Guideline	**Fonte**	**Objetivo**
Definir escopo com base em objetivos do negócio	*Lean Inception*	Delimitar claramente os limites do sistema e o domínio a ser modelado
Selecionar participantes diversos	Entrevistas com gestores + literatura ágil	Incluir múltiplas perspectivas (negócio, desenvolvimento, segurança etc.)
Adaptar formato para o contexto da equipe	Estudo de caso prático	Utilizar ferramentas e práticas adequadas ao contexto, especialmente remoto
Utilizar linguagem ubíqua	DDD + Brandolini	Evitar ambiguidades e promover entendimento comum entre todos os envolvidos
Promover participação ativa dos envolvidos	Estudo de caso + entrevistas	Estimular um ambiente colaborativo onde as contribuições são visíveis e valorizadas
Documentar e atualizar a *Big Picture* após a sessão	Prática recomendada + entrevistas	Garantir persistência e compartilhamento do conhecimento gerado
Transformar o conteúdo gerado em artefatos de engenharia	Prática de modelagem + análise do estudo	Produzir histórias de usuário, estimativas de esforço e planos de *sprint* baseados no modelo
Integrar com outras metodologias e ferramentas	*Lean Inception*, BDD, Scrum, UML	Reforçar a aplicabilidade do *Event Storming* no ciclo completo de desenvolvimento ágil
Incluir elicitação de requisitos não funcionais	Análise da superfície de ataque + entrevistas	Antecipar preocupações com segurança, desempenho, escalabilidade, etc. desde o início

Quadro 11 – *Guidelines* para Adoção do *Event Storming*

5.3.1 Fundamentação das *Guidelines*

A formulação das diretrizes apresentadas na Quadro 11 foi realizada a partir da combinação dos elementos apresentados até o momento: revisão dos trabalhos relacionados, experiências práticas conduzidas pelo autor e entrevistas com gestores atuantes em times ágeis. O objetivo foi reunir orientações práticas para apoiar a adoção do *Event Storming* na elicitação de requisitos, com ênfase na integração com métodos ágeis e na consideração de requisitos não funcionais.

A fundamentação teórica partiu de obras centrais sobre *Event Storming*, especialmente as propostas por Brandolini, além de materiais sobre *Domain-Driven Design* (DDD), *Lean Inception*, *SCRUM*, BDD e UML. Essas referências embasaram diretrizes relativas à estruturação do *workshop*, o uso de uma linguagem ubíqua e a integração com outras práticas.

Paralelamente, a experiência prática de modelagem da superfície de ataque, conduzida pelo autor com base em um sistema real, possibilitou a identificação de como a técnica pode ser adaptada para se antecipar perante vulnerabilidades e tratar este requisito não funcional ainda na fase de design. Esse experimento embasou diretrizes voltadas ao aproveitamento não convencional da técnica, com foco em segurança e desempenho.

Adicionalmente, entrevistas com gestores de projeto revelaram dificuldades recorrentes na aplicação de dinâmicas colaborativas principalmente em grupos numerosos. Com base nesses relatos, foram extraídas diretrizes voltadas à seleção de participantes, à adaptação do formato para contextos remotos e à importância da documentação após a sessão.

Vale destacar que o estudo experimental com alunos, apresentado na próxima seção (5.4), foi conduzido posteriormente à construção das *guidelines*, com o objetivo de avaliar sua eficácia prática em um ambiente educacional controlado.

Dessa forma, as *guidelines* resultantes configuram um artefato fundamentado em múltiplas fontes de evidência e oferecem suporte para a aplicação do *Event Storming* na engenharia de requisitos, especialmente em contextos ágeis.

Seguindo o roteiro do DSR, uma vez que nosso artefato foi produzido, o próximo passo é validá-lo.

5.4 Validando o Artefato : Relato de Experiência com *Survey*

O uso de Modelos de Linguagem de Grande Escala (LLMs) tem ganhado crescente destaque na educação, especialmente em áreas técnicas e complexas como a Engenharia de Requisitos (ARVIDSSON; AXELL, 2023). A capacidade desses modelos de processar e gerar texto com alta precisão oferece novas oportunidades para a construção de conhecimento, proporcionando uma ferramenta poderosa para estudantes e profissionais. Contudo, a popularização dos LLMs também levanta questões sobre a precisão e a veracidade das informações geradas (JACOB; KERRIGAN; BASTOS, 2025), destacando a necessidade de uma abordagem crítica e educativa para maximizar os benefícios dessa tecnologia.

O uso da IA como ferramenta de apoio ao processo de engenharia de requisitos pode ser mais eficiente e preciso, dando a oportunidade de se avaliar o resultado gerado por esta ferramenta para complemento do processo de levantamento de requisitos.

5.4.1 Objetivos e Avaliação

Nesta etapa do estudo visa-se analisar a experiência de introduzir o uso responsável e crítico de LLMs no processo de ensino-aprendizagem de Engenharia de Software. Além disso, o estudo busca avaliar se é possível estabelecer uma relação entre os resultados de sessões de *Event Storming* como insumo para gerar requisitos funcionais e não funcionais por meio da LLM seguindo as *guidelines* apresentadas no Quadro 11.

Com base nesta premissa, este artigo tem como objetivo responder à seguinte pergunta: Como estudantes de Engenharia de Software avaliam o uso de LLMs para geração de requisitos com base em *Event Storming*?

Em caráter auxiliar, novas questões de pesquisa devem ser respondidas com este estudo apresentadas no Quadro 12:

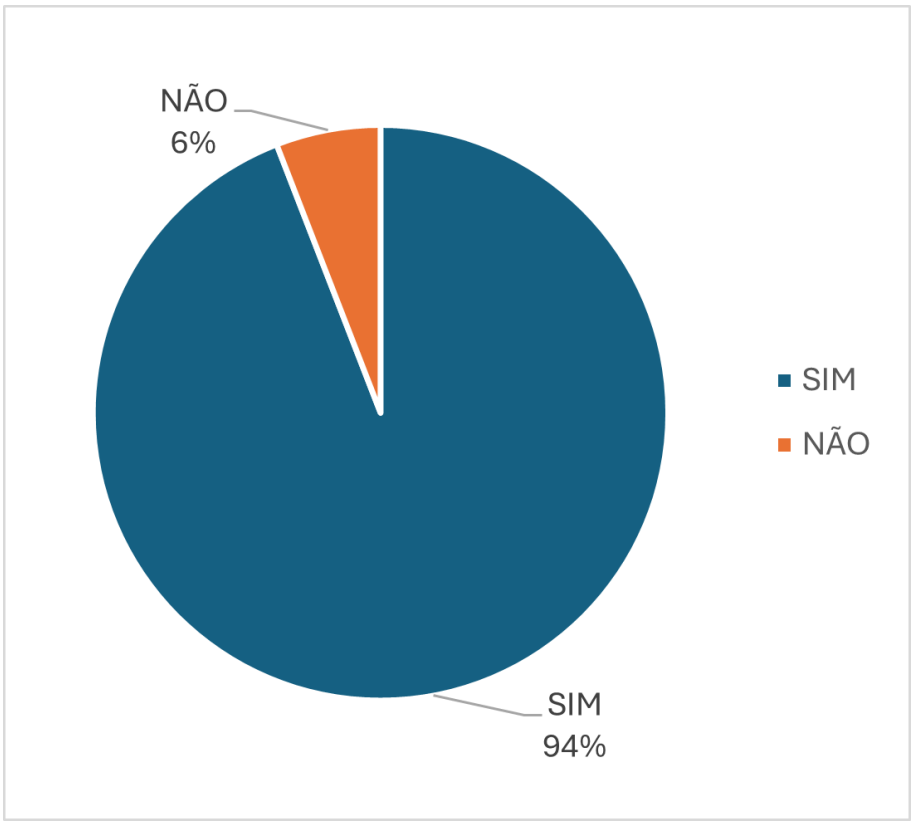
RQ1	Como os estudantes avaliam a qualidade dos requisitos funcionais e não funcionais gerados por LLMs a partir de insumos produzidos via *Event Storming*?
RQ2	A utilização do *Event Storming* contribui para a elaboração de *prompts* mais eficazes para a geração de requisitos por LLMs?
RQ3	O uso combinado de *Event Storming* e LLM promove o desenvolvimento da autonomia crítica dos estudantes na avaliação de requisitos?

Quadro 12 – Questões de pesquisa

A partir da análise descritiva das respostas dos questionários de avaliação individuais, os dados foram agrupados e algumas observações iniciais emergiram

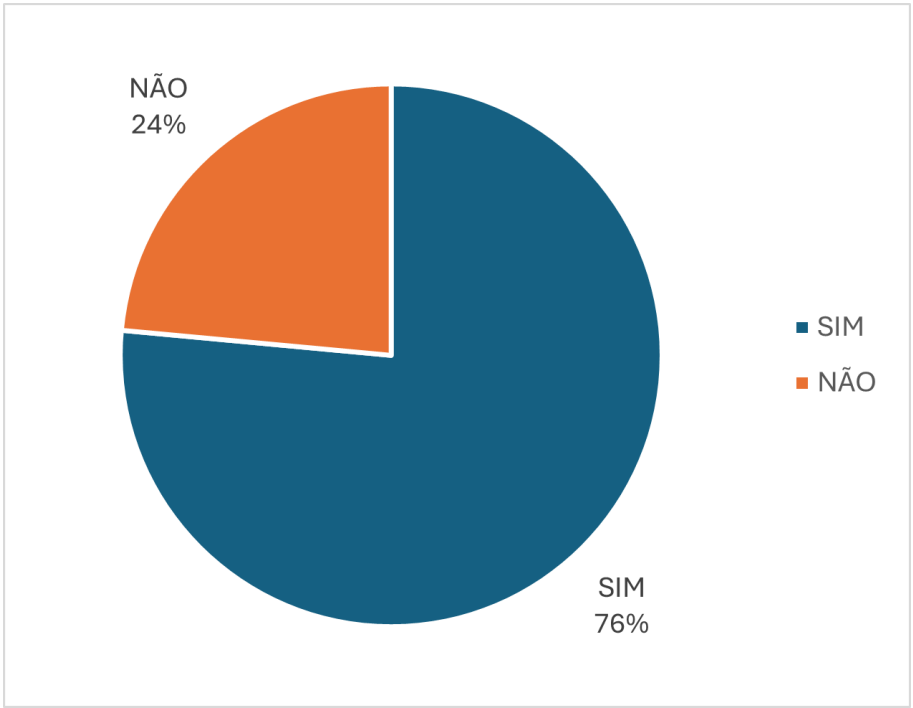
- **Experiência do aluno:** Apenas um aluno relatou ter utilizado a técnica *Event Storming* antes do experimento, embora a maioria já tivesse trabalhado com LLMs para lidar com requisitos de software.

Figura 13 – Q1- É a primeira vez usando *Event Storming* como técnica de requisitos?



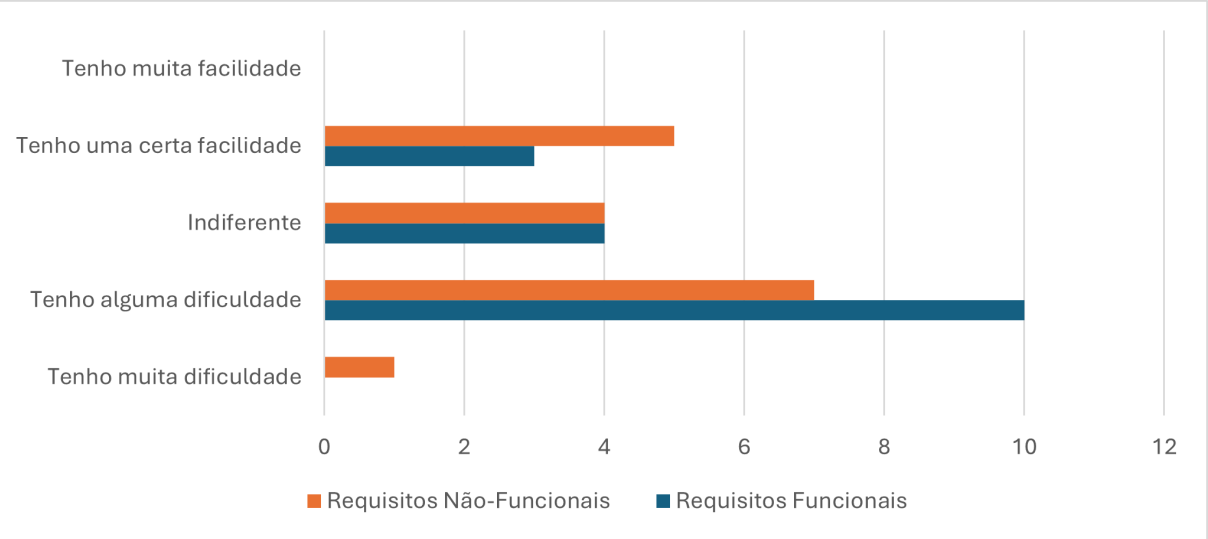
Fonte: Elaborado pelo autor.

Figura 14 – Q2 Já utilizou LLMs (ChatGPT, Gemini, Copilot etc.) para produzir requisitos de *software* anteriormente?



Fonte: Elaborado pelo autor.

Figura 15 – Q3 e Q4 - Como que você avalia sua capacidade de elaborar requisitos funcionais para o seu projeto de *software*?

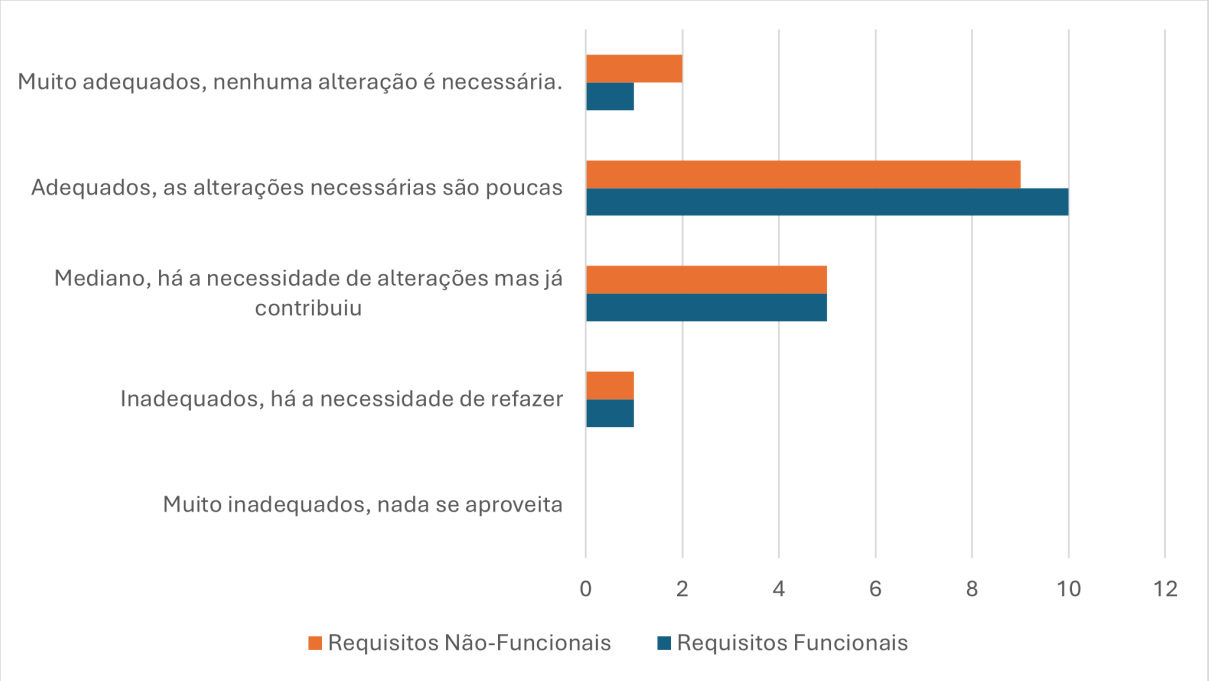


Fonte: Elaborado pelo autor.

- **Competência autoavaliada:** A maioria dos alunos relatou dificuldades na escrita de requisitos. Requisitos funcionais foram mais fáceis de desenvolver do que os não funcionais.
- **Avaliação dos requisitos funcionais e não funcionais gerados pelo LLM:** A maioria dos alunos considerou os requisitos funcionais e não funcionais gerados pelo LLM adequados. No entanto, eles notaram a necessidade de ajustes.
- **Avaliação das recomendações de qualidade:** Embora 42% dos alunos tenham considerado as recomendações relacionadas à qualidade úteis, nenhum aluno expressou críticas negativas a esse aspecto.
- **Reutilização de LLMs para engenharia de requisitos:** Apenas 35,3% dos alunos indicaram que considerariam usar um LLM novamente para geração de requisitos, enquanto 11,8% relataram indiferença.

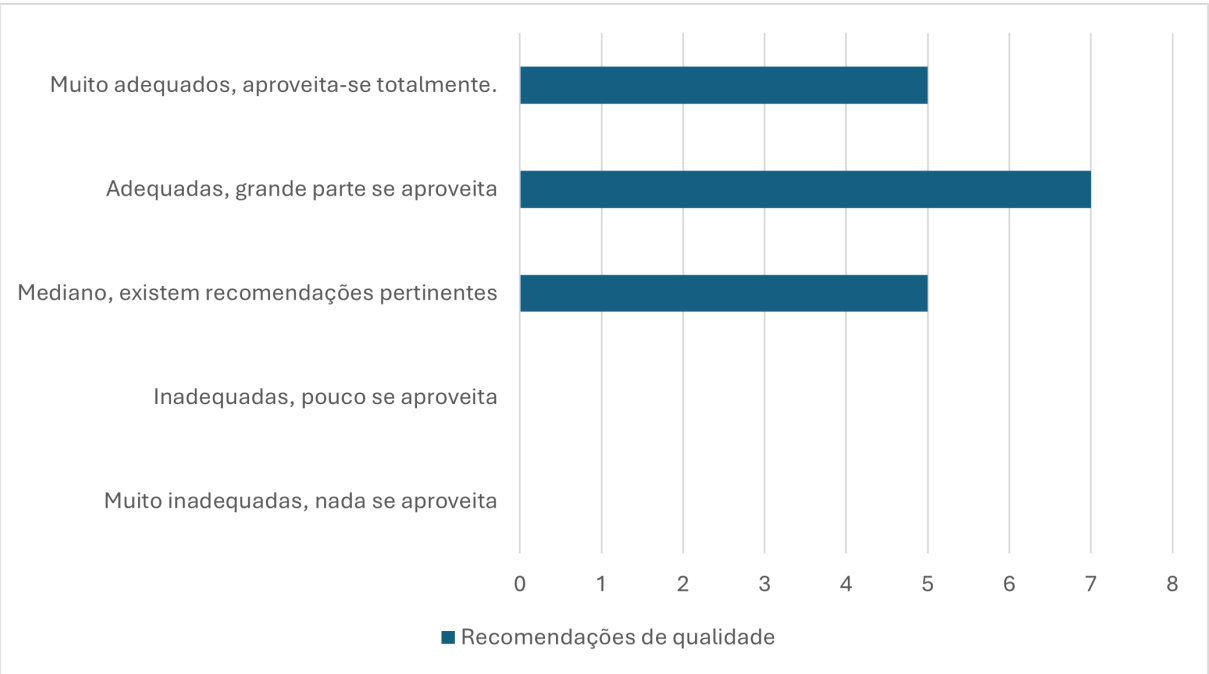
Os dados do questionário geraram diversos gráficos para análise. Quando os alunos foram inicialmente solicitados a desenvolver histórias de usuário livremente, isso deixou espaço para um uso mecânico sem crítica alguma de ferramentas de inteligência artificial. No entanto, ao incorporar explicitamente o LLM à atividade, os alunos foram incentivados a avaliar os resultados de forma crítica. Esse engajamento crítico se reflete nas respostas, conforme ilustrado nas Figuras 16 e 17. Os resultados são ainda mais relevantes, visto que os alunos foram estimulados a refletir sobre sua capacidade de desenvolver requisitos funcionais e não funcionais, como mostrado na Figura 15. Ao comparar a intenção de usar LLMs para gerar requisitos funcionais e não funcionais, fica evidente que, entre os participantes, quanto maior a dificuldade em desenvolver tais requisitos de forma independente, maior o interesse em continuar a usar LLMs como ferramenta de suporte

Figura 16 – Q5 e Q6 - De um modo geral, como você avalia o resultado dos requisitos gerados a partir da LLM?



Fonte: Elaborado pelo autor.

Figura 17 – Q7 - As recomendações para garantir a qualidade do produto foram adequadas para o projeto?



Fonte: Elaborado pelo autor.

Capacidade Funcional	Média da Avaliação da LLM	Desvio Padrão
Tenho muita dificuldade	3.5	0.6
Tenho alguma dificuldade	3.2	0.7
Indiferente	3.0	0.5
Tenho certa facilidade	2.7	0.6
Tenho facilidade	2.2	0.4

Tabela 1 – Relação entre Capacidade Funcional e Avaliação dos resultados obtidos.

— sugerindo que a IA oferece valor compensatório em contextos educacionais. Da mesma forma, ao comparar a qualidade percebida dos requisitos gerados para reutilizar LLMs, surge uma relação entre a utilidade percebida dos resultados e a disposição de confiar em LLMs para geração futura de requisitos. Uma observação importante surge ao comparar a capacidade dos alunos de formular requisitos com sua avaliação dos resultados do LLM: alunos que relatam menos dificuldade em escrever requisitos tendem a ser mais críticos ao avaliar as respostas geradas pela IA.

RQ1 – Como os alunos avaliam a qualidade dos requisitos funcionais e não funcionais gerados pelos LLMs com base nas informações produzidas pelo Event Storming?

Duas perguntas do questionário de avaliação individual (Questões 5 e 6) abordaram a qualidade dos requisitos gerados. Em relação aos requisitos funcionais, 58% dos alunos afirmaram que os resultados foram adequados, embora alguns ajustes tenham sido necessários. Além disso, 29,4% consideraram os resultados médios, mas potencialmente úteis para a análise de requisitos. Para os requisitos não funcionais, os resultados foram semelhantes: 52,9% os consideraram adequados, com algumas ressalvas, enquanto 29,4% consideraram os resultados médios, mas utilizáveis. Diante dessas constatações, os alunos, em geral, consideraram os requisitos gerados pelo LLM como positivos e valiosos.

RQ2 – O uso do Event Storming contribui para a criação de prompts *mais eficazes para a geração de requisitos pelos LLMs?*

Mesmo sem experiência prévia com *Event Storming*, a maioria dos alunos obteve resultados satisfatórios. O uso de *prompts*, combinado com o *Big Picture*, sugere que representações visuais de eventos e comandos podem ser efetivamente empregadas para apoiar a geração de requisitos pelo LLM.

RQ3 – O uso combinado de Event Storming e LLMs promove o desenvolvimento da autonomia crítica dos alunos na avaliação de requisitos?

O experimento foi projetado para incluir uma fase de avaliação integrada dos resultados gerados. Em vez de simplesmente copiar as saídas produzidas pelo LLM, os alunos foram solicitados a compará-las com os requisitos que eles próprios construíram. Esse processo permitiu que os alunos avaliassem a qualidade das saídas geradas pela IA com base em sua capacidade de formular requisitos, levando-os a identificar quais partes precisavam de melhorias.

6
Discussão

Todos os estudos conduzidos até aqui merecem uma avaliação crítica para que possam se consolidar como um importante passo para o futuro desta pesquisa. A partir da análise do resultado do uso das *Guidelines* pelos estudantes em conjunto com a utilização de LLM como um processo final, esta seção busca evidenciar avanços, soluções e limitações do estudo como um todo.

6.1 Análise Crítica das *Guidelines*

Conforme mencionado na seção 5.3, as *guidelines* propostas foram construídas a partir do que já estava disponível na literatura, na entrevista com os gestores e no estudo de caso de análise de superfície de ataque.

Essas *guidelines* têm como propósito estabelecer um conjunto de boas práticas para o uso do *Event Storming*, tornando-o mais acessível e replicável por equipes multidisciplinares com diferentes níveis de experiência.

Estas *guidelines* foram submetidas à avaliação prática com os estudantes e demonstraram que elas facilitaram a execução da dinâmica e tornaram todo o processo mais transparente, aumentando assim a compreensão dos participantes. Vale ressaltar que mesmo os estudantes com menos experiência puderam se beneficiar dos resultados após seguirem as *guidelines*, o que corrobora a efetividade da proposta.

No entanto, foram observadas limitações. Algumas *guidelines* exigem conhecimento prévio em modelagem ou experiência para que possam ser executadas com mais facilidade, o que pode ser uma barreira em contextos menos técnicos. Além disso, o ambiente acadêmico em si, embora seja bastante valioso, está mais para um ambiente simulado, o que pode não refletir todas as complexidades de um time real de desenvolvimento ágil.

Comparando com abordagens tradicionais como a UML, as *guidelines* propostas mantêm um grau de informalidade que favorece a integração com métodos ágeis. Nesse sentido, as *guidelines* avançam sobre a literatura no uso do *Event Storming* como ferramenta de documentação visual orientada a requisitos.

6.2 Uso do *Event Storming* para Documentação de Segurança

Os estudos preliminares à construção das *guidelines*, mais precisamente no de Análise de Superfície de Ataque, permitiram entender como o *Event Storming* contribui para identificar potenciais pontos vulneráveis e expostos do sistema. A partir da interação dos elementos de domínio, foi possível identificar vetores de ataque por meio da identificação de eventos e comandos.

Ao passo que ferramentas de modelagem mais tradicionais como UML e DFD são utilizadas para este fim, o *Event Storming* oferece uma solução com poucos formalismos e pode ser efetuada antes da definição da arquitetura, ou seja, nas fases iniciais de design. Ainda que seja necessária uma ferramenta de modelagem formal, nada impede de se utilizar o *Event Storming* como ferramenta inicial de apoio ao antecipar os riscos de segurança.

6.3 Uso de LLM como Apoio à Elicitação de Requisitos

Na experiência envolvendo os estudantes, o potencial do uso de LLM para obtenção de requisitos funcionais e não funcionais foi observado. A partir da *Big Picture*, os resultados foram gerados e bem avaliados pelos próprios estudantes. O uso da IA mostrou-se útil ao interpretar os comandos e eventos em linguagem ubíqua, fornecendo bons resultados.

O maior benefício do uso da LLM na geração de requisitos a partir de uma *Big Picture* se deu na superação da barreira inicial da formulação dos primeiros requisitos.

Apesar disso, o uso da IA requer o máximo de atenção. Em alguns casos, a LLM gerou informações imprecisas e até mesmo falhas, exigindo que os estudantes validassem criticamente as saídas. Isso indica que, embora bastante úteis, as LLMs não substituem o papel do ser humano, mas sim o ampliam, desde que os usuários estejam preparados para avaliar os resultados.

6.4 Limitações da Pesquisa

Dentre as limitações deste estudo, destacam-se o escopo utilizado para a validação do artefato. Embora tenha sido muito promissor o uso do contexto acadêmico, os participantes não possuem experiência real com projetos, limitando assim o estudo. Outro ponto limitador foi a indisponibilidade de avaliação dos resultados pelos gestores entrevistados no estudo pré-construção do artefato, o que limita a avaliação da aplicabilidade em um projeto real. Tal limitação

não invalida o artefato uma vez que o *Design Science Research* prevê a validação de seus artefatos em ambientes simulados, porém nos convida a expandir a validação para um contexto de projeto real ainda que em menor escala.

Futuros estudos podem explorar o uso das *guidelines* em ambientes empresariais, com equipes multidisciplinares e prazos reais. Além disso, por conta dos resultados do uso de LLM terem sido bem aceitos, sugere-se o desenvolvimento de conjuntos de *prompts* validados para uso com estas LLMs.

6.5 Avanços Conceituais e Práticos

Combinar *Event Storming* com as *guidelines* propostas e o uso de IA representam um avanço na Engenharia de Requisitos. Esta proposta oferece uma abordagem acessível, de baixo custo, colaborativa, alinhada com as práticas de metodologias ágeis e amplia o portfólio de técnicas disponíveis para os analistas de requisitos.

Este estudo também demonstra a contribuição de IA com a integração de técnicas que oferecem artefatos visuais, contribuindo não apenas com requisitos, mas também com a formação crítica dos profissionais desde o contexto acadêmico.

Além desses benefícios conceituais, a aplicação prática das *guidelines* demonstrou ganhos no uso do *Event Storming* para microsserviços. O método estruturado sugerido pelas *guidelines* facilitou a condução das sessões, reduziu o tempo de documentação e favoreceu a identificação de requisitos e pontos críticos de segurança ainda na fase de design. A integração com LLMs fomentou a transformação das *Big Pictures* em requisitos funcionais e não funcionais claros, mantendo consistência terminológica e reduzindo retrabalho.

Também se comprovou a aplicabilidade das *guidelines* em contextos diversos, desde novos projetos até iniciativas de reengenharia, com boa aceitação por equipes técnicas e gerenciais, e viabilidade de uso em ambientes presenciais, remotos e híbridos conforme experimento com os alunos.

Ainda há de se notar a utilização efetiva das *guidelines* para propor o uso do *Event Storming*, uma técnica colaborativa, na construção da análise de superfície, diferente de métodos tradicionais que não favorecem o debate sobre o assunto enquanto a documentação é construída.

A avaliação do artefato demonstrou sua eficácia ao viabilizar o uso das *Big Pictures* em conjunto com LLM, uma vez que os participantes conseguiram gerar requisitos funcionais e não funcionais com boa qualidade a partir dos artefatos visuais da técnica. A atividade favoreceu o pensamento crítico e a autonomia dos envolvidos, e a presença das *guidelines* funcionou como um guia para estruturar a dinâmica e melhorar a qualidade dos *prompts*, reduzindo respostas genéricas ou equivocadas da IA. Isso reforça a relevância do papel das *guidelines* como instrumento pedagógico e metodológico.

7

Considerações Finais

A partir dos resultados obtidos ao longo desta pesquisa, foi possível demonstrar com êxito a eficácia das *guidelines* propostas para a adoção do *Event Storming* como ferramenta adequada para a etapa de análise de requisitos em contextos de times ágeis. As *guidelines* formuladas consolidam-se como um artefato útil, com bases fundamentadas tanto na literatura quanto em evidências empíricas obtidas por meio de experimentos, entrevistas e estudo de caso.

A aplicação prática da técnica, aliada ao uso de uma ferramenta de inteligência artificial generativa, evidenciou que mesmo entre os participantes sem familiaridade com o *Event Storming* tivemos a capacidade de observar a condução de sessões produtivas. As *guidelines* atuaram como um suporte sólido, facilitando a preparação, condução e documentação da dinâmica. Como resultado, observou-se a geração de artefatos mais completos, contextualizados e alinhados com o domínio do negócio, o que favoreceu a participação ativa e a compreensão dos envolvidos.

Adicionalmente à combinação entre *Event Storming* e IA generativa, revelou-se promissora. A IA contribuiu de maneira efetiva na validação dos artefatos gerados por sessões de *Event Storming*, fornecendo sugestões de histórias de usuários relevantes, assim como recomendações de qualidade. Essa integração, quando mediada por boas práticas e orientações pertinentes, amplia o potencial da técnica como ferramenta acessível e replicável para diferentes públicos e cenários.

Do ponto de vista metodológico, a pesquisa foi guiada pelos princípios do *Design Science Research*, ao propor, desenvolver e avaliar um artefato que tem como objetivo contribuir para a prática da engenharia de requisitos. As *guidelines* respondem a uma lacuna identificada na literatura, onde há escassez de orientações sistematizadas sobre como empregar o *Event Storming* de forma eficaz em contextos reais nos times ágeis.

Contudo, todo o estudo conduzido até aqui jamais poderia ser considerado definitivo haja vista algumas limitações que merecem ser reconhecidas. A principal delas refere-se à natureza dos ambientes de aplicação: os experimentos ocorreram em cenários acadêmicos ou simulados,

o que pode não refletir plenamente a complexidade de projetos reais em ambientes corporativos. Além disso, o uso de IA generativa ainda exige mediação humana que possua um nível de crítica para garantir resultados consistentes e contextualmente relevantes.

Do ponto de vista metodológico, a utilização do Design Science Research garantiu a construção, avaliação e validação do artefato, evidenciando que as guidelines são tanto uma contribuição teórica quanto prática. Os resultados destacaram que mesmo em ambientes acadêmicos, a técnica promoveu sessões produtivas, participação ativa dos envolvidos e reflexões críticas sobre o papel da IA no apoio à engenharia de requisitos.

A eficácia das *guidelines* propostas foi evidenciada no terceiro estudo, o relato de experiência com *survey*. Nele, os participantes aplicaram integralmente o processo definido, realizando sessões de *Event Storming* e, em seguida, utilizando LLMs para transformar as *Big Pictures* em requisitos funcionais e não funcionais consistentes. As respostas ao questionário indicaram um *feedback* positivo quanto à utilidade das *guidelines*, incrementando a confiança na elaboração de requisitos e intenção de reutilização da técnica.

A análise quantitativa dos resultados indica que 76% dos participantes avaliaram os requisitos funcionais gerados como "adequados" ou "muito adequados", enquanto 65% atribuíram essa mesma classificação aos requisitos não funcionais. Apesar de parte dos respondentes relatar "alguma dificuldade" na elaboração de requisitos, mais da metade declarou ao menos "certa facilidade" após a experiência.

Em relação às recomendações de qualidade, 80% consideraram-nas "adequadas" ou "muito adequadas" para garantir a qualidade do produto. Além disso, a intenção de uso futuro demonstrou-se promissora: 82% afirmaram que voltariam a utilizar LLMs para gerar requisitos funcionais e não funcionais de maneira assistida.

Esses resultados reforçam a aplicabilidade prática e o potencial de expansão da abordagem para contextos acadêmicos e profissionais, com destaque para a formação de novos analistas e o apoio a equipes ágeis, promovendo maior agilidade, padronização documental e efetividade na comunicação entre *stakeholders*.

Do ponto de vista pedagógico, a atividade estimulou a autonomia crítica dos estudantes, que não apenas consumiram passivamente os resultados, mas os analisaram e refletiram sobre sua aplicabilidade, limitações e pontos de revisão. Essa abordagem combinada se mostrou eficaz para promover o pensamento crítico acerca do uso de ferramentas de IA generativa, fortalecendo sua utilidade pedagógica na formação em Engenharia de Software.

Podemos de fato concluir que a combinação de métodos colaborativos e visuais, como o *Event Storming*, com ou sem o suporte de tecnologias baseadas em IA, representa um caminho promissor para modernizar e tornar mais inclusivo o processo de engenharia de requisitos. As *guidelines* desenvolvidas neste trabalho oferecem um avanço nessa direção, promovendo a integração, inovação e aplicabilidade, com impacto na formação acadêmica e na atuação

profissional em Engenharia de Software, além de expandir o contexto de aplicabilidade do *Event Storming*.

7.1 Síntese das Respostas às Questões de Pesquisa

Os objetivos estabelecidos no início desta pesquisa nortearam toda a condução do estudo. O objetivo geral consistiu em avaliar a aplicabilidade do *Event Storming* na definição de requisitos funcionais e não funcionais em uma arquitetura de microsserviços. Para isso, foram definidos três objetivos específicos: (i) avaliar a capacidade de utilização do *Event Storming* como ferramenta de análise de requisitos, (ii) produzir *guidelines* para aplicação da técnica em contextos ágeis e (iii) avaliar a eficácia dessas *guidelines* por meio de uma aplicação técnica apoiada por Inteligência Artificial .

Com base nesses objetivos, foi possível responder às perguntas de pesquisa. Em relação à P1 (“Como o *Event Storming* pode ser utilizado de forma eficaz como ferramenta para análise de requisitos em contextos de times ágeis?”), verificou-se que o *Event Storming* é uma eficaz ferramenta para análise de requisitos, especialmente em contextos ágeis, por promover uma compreensão compartilhada do domínio além de facilitar a comunicação entre os envolvidos. A técnica se mostrou acessível e capaz de estruturar a descoberta de eventos e comandos relevantes dos projetos de sistemas. A aplicação prática reforçou seu potencial como instrumento de elicitação e análise colaborativa e visual, superando limitações de abordagens tradicionais baseadas unicamente em documentação formal, uma vez que se apresenta na forma de linguagem ubiqua. .

Quanto à P2 (“Quais *guidelines* podem ser formuladas para orientar a aplicação do *Event Storming* como técnica de elicitação e análise de requisitos?”),Foi possível formular um conjunto de *guidelines* que orientam a aplicação do *Event Storming* com foco especial na análise de requisitos. As *guidelines* abrangem aspectos como preparação do ambiente, definição clara dos papéis e apoio de ferramentas visuais e tecnológicas. Tais *guidelines* foram organizadas de maneira a viabilizar a replicabilidade da prática e a facilitar sua adoção tanto em ambientes ágeis como em contextos educacionais.

Já em relação à P3 (“As *guidelines* propostas contribuem efetivamente para a aplicação do *Event Storming* com suporte de inteligência artificial generativa na geração de requisitos de software?”), a avaliação do artefato demonstrou sua eficácia ao viabilizar o uso das Big Pictures em conjunto com LLM, uma vez que os participantes conseguiram gerar requisitos funcionais e não funcionais com boa qualidade a partir dos artefatos visuais da técnica. A atividade favoreceu o pensamento crítico e a autonomia dos envolvidos, e a presença das guidelines funcionou como um guia para estruturar a dinâmica e melhorar a qualidade dos prompts, reduzindo respostas genéricas ou equivocadas da IA. Isso reforça a relevância do papel das guidelines como instrumento pedagógico e metodológico.

7.2 *Roadmap* de Pesquisa

Como projetos para o futuro, a realização de estudos em organizações de software que adotem as *guidelines* aqui propostas, bem como a adaptação do artefato para outros níveis de maturidade de software. Há também espaço para a expansão das próprias *guidelines*, incorporando recomendações específicas para a integração com ferramentas automatizadas e abordagens voltadas à segurança desde o design.

Sendo assim, foi elaborado um *roadmap* de pesquisa que visa maturar o artefato para que possa aderir por completo aos ambientes reais de produção.

Aplicar as guidelines em novos estudos de caso, abrangendo diferentes domínios de software: replicar a metodologia em grupos maiores; comparar desempenho entre equipes que usaram com as que não usaram as guidelines; Analisar métricas mais objetivas e de qualidade de software.

Avaliar como as guidelines se adaptam a modelos de linguagem distintos.: Comparar os resultados de LLM diferentes (comerciais e open source); avaliar taxas de alunicinação; Ajustar as *guidelines* para modelos de linguagem distintos.

Incorporar as guidelines em ferramentas digitais que auxiliem o processo.: desenvolver um protótipo de assistente interativo que guie o analista na aplicação das *guidelines*; testar a integração com plataformas de gestão de requisitos (Jira, Trello, Azure DevOps); medir impacto na produtividade e satisfação dos usuários.

Medir o impacto da aplicação das guidelines no longo prazo.: acompanhar equipes que usem as *guidelines* durante todo o ciclo de desenvolvimento ; Avaliar a redução de retrabalho e mudanças de requisitos, manutenção e evolução do *software*.

Garantir aplicabilidade em contextos reais e comerciais.: conduzir pilotos em empresas de diferentes portes e analisar viabilidade econômica da adoção.

Referências

AMNA, A. R.; POELS, G. Ambiguity in user stories: A systematic literature review. *Information and Software Technology*, v. 145, p. 106824, 2022. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584922000040>. Citado na página 50.

ARORA, C.; GRUNDY, J.; ABDELRAZEK, M. Advancing requirements engineering through generative ai: Assessing the role of llms. In: *Generative AI for Effective Software Development*. [S.l.]: Springer, 2024. p. 129–148. Citado 2 vezes nas páginas 28 e 30.

ARVIDSSON, S.; AXELL, J. Prompt engineering guidelines for llms in requirements engineering. 2023. Citado na página 55.

BALALAIE, A.; HEYDARNOORI, A.; JAMSHIDI, P. Migrating to cloud-native architectures using microservices: an experience report. In: SPRINGER. *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15-17, 2015, Revised Selected Papers 4*. [S.l.], 2016. p. 201–215. Citado na página 22.

BELLOVIN, S. M. Attack surfaces. *IEEE Security & Privacy*, IEEE, v. 14, n. 3, p. 88–88, 2016. Citado na página 23.

BELZNER, L.; GABOR, T.; WIRSING, M. Large language model assisted software engineering: prospects, challenges, and a case study. In: SPRINGER. *International conference on bridging the gap between AI and reality*. [S.l.], 2023. p. 355–374. Citado 2 vezes nas páginas 28 e 30.

BRADBURY, M. et al. Identifying attack surfaces in the evolving space industry using reference architectures. In: IEEE. *2020 IEEE Aerospace Conference*. [S.l.], 2020. p. 1–20. Citado 3 vezes nas páginas 27, 29 e 43.

BRANDOLINI, A. Introducing eventstorming: An act of deliberate collective learning. *Leanpub: Victoria, BC, Canada*, 2018. Citado 6 vezes nas páginas 14, 19, 23, 25, 34 e 44.

CAROLI, P. Lean inception: how to align people and build the right product. *Editora Caroli*, 2018. Citado 2 vezes nas páginas 18 e 38.

CAROLI, P. *When to do a Lean Inception? - Caroli.org*. 2022. <https://caroli.org/en/when-to-do-a-lean-inception/>. (Accessed on 05/27/2024). Citado 3 vezes nas páginas 48, 50 e 52.

CARRANZA-GARCÍA, F.; RODRÍGUEZ-DOMÍNGUEZ, C.; GARRIDO, J. L. Addressing expressiveness for a uml microservices-based modeling within the life cycle of the ubiquitous system development. In: IEEE. *2021 17th International Conference on Intelligent Environments (IE)*. [S.l.], 2021. p. 1–8. Citado na página 18.

CHEN, C.-A. With great abstraction comes great responsibility: Sealing the microservices attack surface. In: IEEE. *2019 IEEE Cybersecurity Development (SecDev)*. [S.l.], 2019. p. 144–144. Citado 2 vezes nas páginas 14 e 23.

CHRISTOFOROU, A. et al. Supporting the decision of migrating to microservices through multi-layer fuzzy cognitive maps. In: SPRINGER. *Service-Oriented Computing: 15th*

International Conference, ICSOC 2017, Malaga, Spain, November 13–16, 2017, Proceedings. [S.l.], 2017. p. 471–480. Citado 2 vezes nas páginas 22 e 23.

DAFFODIL-SOFTWARE. *Monolithic vs Microservices: Which is the Better Architecture for eCommerce App Development*. 2020. Disponível em: <https://daffodilsw.medium.com/monolithic-vs-microservices-which-is-the-better-architecture-for-ecommerce-app-development-47f7466223 Acesso em: 05 set. 2025. Citado na página 35.

DAWEI, Y. et al. Design and achievement of security mechanism of api gateway platform based on microservice architecture. *Journal of Physics: Conference Series*, IOP Publishing, v. 1738, n. 1, p. 012046, jan 2021. Disponível em: <https://dx.doi.org/10.1088/1742-6596/1738/1/012046>. Citado na página 41.

EDISON, H.; WANG, X.; CONBOY, K. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, IEEE, v. 48, n. 8, p. 2709–2731, 2021. Citado 2 vezes nas páginas 24 e 25.

EVANS, E. *Domain-driven design: tackling complexity in the heart of software*. [S.l.]: Addison-Wesley Professional, 2004. Citado 2 vezes nas páginas 14 e 22.

GARDNER, H.; BLACKWELL, A. F.; CHURCH, L. The patterns of user experience for sticky-note diagrams in software requirements workshops. *Journal of Computer Languages*, Elsevier, v. 61, p. 100997, 2020. Citado na página 44.

GEORG, G. et al. Verification and trade-off analysis of security properties in uml system models. *IEEE Transactions on Software Engineering*, IEEE, v. 36, n. 3, p. 338–356, 2010. Citado 2 vezes nas páginas 27 e 29.

HERRMANN, P. Information flow analysis of component-structured applications. In: IEEE. *Seventeenth Annual Computer Security Applications Conference*. [S.l.], 2001. p. 45–54. Citado 3 vezes nas páginas 26, 29 e 43.

HEVNER, A. R. A three cycle view of design science research. *Scandinavian journal of information systems*, v. 19, n. 2, p. 4, 2007. Citado na página 31.

HOFER, S.; SCHWENTNER, H. *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software*. [S.l.]: Addison-Wesley Professional, 2021. Citado 2 vezes nas páginas 22 e 44.

JACOB, C.; KERRIGAN, P.; BASTOS, M. The chat-chamber effect: Trusting the ai hallucination. *Big Data & Society*, SAGE Publications Sage UK: London, England, v. 12, n. 1, p. 20539517241306345, 2025. Citado na página 55.

JUNKER, A. Microservices as architectural style. *Digitalization in Healthcare: Implementing Innovation and Artificial Intelligence*, Springer, p. 177–190, 2021. Citado na página 14.

JÜRJENS, J.; SCHRECK, J.; BARTMANN, P. Model-based security analysis for mobile communications. In: *Proceedings of the 30th international conference on Software engineering*. [S.l.: s.n.], 2008. p. 683–692. Citado 3 vezes nas páginas 26, 29 e 43.

KAPFERER, S.; ZIMMERMANN, O. Domain-driven service design: Context modeling, model refactoring and contract generation. In: SPRINGER. *Service-Oriented Computing: 14th Symposium and Summer School on Service-Oriented Computing, SummerSOC 2020, Crete, Greece, September 13-19, 2020 14*. [S.l.], 2020. p. 189–208. Citado na página 23.

LAOYAN, S. *Entendendo o princípio de Pareto (a regra 80/20*. 2022. <https://asana.com/pt/resources/pareto-principle-80-20-rule>. (Accessed on 30/08/2023). Citado na página 21.

LARMAN, C. et al. *Applying UML and patterns*. [S.l.]: Prentice Hall Upper Saddle River, 1998. v. 2. Citado na página 18.

LAURETIS, L. D. From monolithic architecture to microservices architecture. In: IEEE. *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. [S.l.], 2019. p. 93–96. Citado na página 22.

LEVCOVITZ, A.; TERRA, R.; VALENTE, M. T. Towards a technique for extracting microservices from monolithic enterprise systems. *ArXiv*, abs/1605.03175, 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:13549058>. Citado 2 vezes nas páginas 26 e 29.

LIU, X. et al. Research on attack mechanism using attack surface. In: IEEE. *2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*. [S.l.], 2020. p. 137–141. Citado na página 42.

MERSON, P.; YODER, J. Modeling microservices with ddd. In: IEEE. *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. [S.l.], 2020. p. 7–8. Citado na página 22.

MOSHTARI, S.; OKUTAN, A.; MIRAKHORLI, M. A grounded theory based approach to characterize software attack surfaces. In: *Proceedings of the 44th International Conference on Software Engineering*. [S.l.: s.n.], 2022. p. 13–24. Citado na página 23.

NEWMAN, S. *Monolith to microservices: evolutionary patterns to transform your monolith*. [S.l.]: O'Reilly Media, 2019. Citado na página 22.

OZKAYA, M. *Padrão de gateway de API | Medium*. 2022. <https://medium.com/design-microservices-architecture-with-patterns/api-gateway-pattern-8ed0ddfce9df#>. (Accessed on 25/05/2023). Citado 2 vezes nas páginas 40 e 41.

PERRY, D. E.; SIM, S. E.; EASTERBROOK, S. Case studies for software engineers. In: *Proceedings of the 28th international conference on software engineering*. [S.l.: s.n.], 2006. p. 1045–1046. Citado 2 vezes nas páginas 31 e 33.

PETRASCH, R. Model-based engineering for microservice architectures using enterprise integration patterns for inter-service communication. In: IEEE. *2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. [S.l.], 2017. p. 1–4. Citado na página 18.

PIMENTEL, M.; FILIPPO, D.; SANTOS, T. M. dos. Design science research: pesquisa científica atrelada ao design de artefatos. *RE@ D-Revista de Educação a Distância e eLearning*, v. 3, n. 1, p. 37–61, 2020. Citado na página 31.

RESELMAN, B. *5 design principles for microservices | Red Hat Developer*. 2022. <https://developers.redhat.com/articles/2022/01/11/5-design-principles-microservices#>. (Accessed on 05/05/2023). Citado 4 vezes nas páginas 26, 29, 34 e 40.

RESELMAN, B. *The disadvantages vs. benefits of microservices | Red Hat Developer*. 2022. <https://developers.redhat.com/articles/2022/01/25/disadvantages-microservices>. (Accessed on 05/05/2023). Citado 2 vezes nas páginas 26 e 29.

RESELMAN, B. *Implementing and using microservices | Red Hat Developer*. 2022. <https://developers.redhat.com/articles/2022/01/19/monolith-microservices-how-applications-evolve#creating_a_microservices_oriented_application>. (Accessed on 05/05/2023). Citado 2 vezes nas páginas 26 e 29.

REZAPOUR, R. et al. Security in fog computing: A systematic review on issues, challenges and solutions. *Computer Science Review*, Elsevier, v. 41, p. 100421, 2021. Citado 2 vezes nas páginas 26 e 29.

ROCHA, F. G.; MISRA, S.; SOARES, M. S. Guidelines for future agile methodologies and architecture reconciliation for software-intensive systems. *Electronics*, v. 12, n. 7, 2023. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/12/7/1582>. Citado 3 vezes nas páginas 15, 24 e 25.

ROCHA, F. G.; SOARES, M. S.; RODRIGUEZ, G. Patterns in microservices-based development: A grey literature review. In: SBC. *Anais do XXVI Congresso Ibero-Americano em Engenharia de Software*. [S.l.], 2023. p. 61–76. Citado na página 22.

RODRIGUEZ, G. et al. Understanding and addressing the allocation of microservices into containers: A review. *IETE Journal of Research*, Taylor & Francis, v. 0, n. 0, p. 1–14, 2023. Disponível em: <https://doi.org/10.1080/03772063.2023.2205864>. Citado na página 14.

ROMAN, R.; LOPEZ, J.; MAMBO, M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, Elsevier, v. 78, p. 680–698, 2018. Citado 2 vezes nas páginas 26 e 29.

RUIZ, G. M.; PANADERO, A. Y. Streamlining product inception and backlog management with large language models. *Available at SSRN 5036454*, 2024. Citado 2 vezes nas páginas 28 e 30.

RUMPE, B. Agile modeling with the uml. In: SPRINGER. *International Workshop on Radical Innovations of Software and Systems Engineering in the Future*. [S.l.], 2002. p. 297–309. Citado na página 18.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, Springer, v. 14, p. 131–164, 2009. Citado 2 vezes nas páginas 33 e 39.

SAIDI, M.; TISSAOUI, A.; FAIZ, S. A ddd approach towards automatic migration to microservices. In: IEEE. *2023 IEEE International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*. [S.l.], 2023. p. 01–06. Citado na página 22.

SANTOS, N. et al. A logical architecture design method for microservices architectures. In: *Proceedings of the 13th European Conference on Software Architecture-Volume 2*. [S.l.: s.n.], 2019. p. 145–151. Citado na página 17.

SIRICHAROEN, W. V. Using empathy mapping in design thinking process for personas discovering. In: SPRINGER. *International Conference on Context-Aware Systems and Applications*. [S.l.], 2020. p. 182–191. Citado 2 vezes nas páginas 19 e 35.

THEISEN, C. et al. Risk-based attack surface approximation: how much data is enough? In: IEEE. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. [S.l.], 2017. p. 273–282. Citado na página 42.

TORKURA, K. A. et al. Securing cloud storage brokerage systems through threat models. In: IEEE. *2018 IEEE 32nd international conference on advanced information networking and applications (AINA)*. [S.l.], 2018. p. 759–768. Citado 2 vezes nas páginas 27 e 30.

TRAN, A.-D.; YSKOUT, K.; JOOSEN, W. Andras: Automated attack surface extraction for android applications. In: IEEE. *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. [S.l.], 2023. p. 406–417. Citado na página 27.

ULUDAĞ, Ö. et al. Supporting large-scale agile development with domain-driven design. In: SPRINGER. *Agile Processes in Software Engineering and Extreme Programming: 19th International Conference, XP 2018, Porto, Portugal, May 21–25, 2018, Proceedings 19*. [S.l.], 2018. p. 232–247. Citado 2 vezes nas páginas 24 e 25.

VERNON, V. *Implementing domain-driven design*. [S.l.]: Addison-Wesley, 2013. Citado na página 22.

VICTOROVA, D. P.; OLEYNIK, P. P. Metamodel-driven design of microservice architecture applications. In: IEEE. *2024 Conference of Young Researchers in Electrical and Electronic Engineering (ElCon)*. [S.l.], 2024. p. 298–301. Citado na página 18.

WEERASINGHE, S.; PERERA, I. Optimized strategy for inter-service communication in microservices. *International Journal of Advanced Computer Science and Applications*, Science and Information (SAI) Organization Limited, v. 14, n. 2, 2023. Citado 2 vezes nas páginas 14 e 34.

WOJCZAL, M. *How to rapidly deliver an Agile project with Event Storming and Design System. Case Study. | LinkedIn*. 2023. <https://www.linkedin.com/pulse/how-rapidly-deliver-agile-project-event-storming-design-wojczal/>. (Accessed on 11/08/2023). Citado 2 vezes nas páginas 24 e 25.

YEE, G. Reducing the attack surface for private data. In: *Proc. Thirteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2019)*. [S.l.: s.n.], 2019. p. 28–34. Citado 2 vezes nas páginas 27 e 30.

ZHANG, W. Security monitoring of microservice-based applications. TechRxiv, 2023. Citado 3 vezes nas páginas 26, 29 e 42.