UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I - CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EDILSON DO NASCIMENTO COSTA JÚNIOR

UMA CONTRIBUIÇÃO PARA A ESPECIFICAÇÃO DE REQUISITOS DE SEGURANÇA EM CONTEXTO ÁGIL

CAMPINA GRANDE - PB
2025

EDILSON DO NASCIMENTO COSTA JÚNIOR

UMA CONTRIBUIÇÃO PARA A ESPECIFICAÇÃO DE REQUISITOS DE SEGURANÇA EM CONTEXTO ÁGIL

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharela em Computação.

Área de concentração: Engenharia de Software

Orientador: Profa. Dra. ANA ISABELLA MUNIZ LEITE

CAMPINA GRANDE - PB

2025

C837c Costa Júnior, Edilson do Nascimento.
Uma contribuição para a especificação de requisitos de segurança em contexto ágil [manuscrito] / Edilson do Nascimento Costa Júnior. - 2025.
72 f. : il. color.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Ciência da computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia, 2025.

"Orientação : Prof. Dra. Ana Isabella Muniz Leite, Departamento de Computação - CCT".

1. Segurança da informação. 2. Sistemas de saúde. 3. Requisitos de segurança. I. Título

21. ed. CDD 005.1

Elaborada por Bruno Rafael Freitas de Lima - CRB - 15/1021 BC

EDILSON DO NASCIMENTO COSTA JUNIOR

UMA CONTRIBUIÇÃO PARA A ESPECIFICAÇÃO DE REQUISITOS DE SEGURANÇA EM CONTEXTO ÁGIL

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciência da Computação da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de Bacharel em Computação

Aprovada em: 11/06/2025.

BANCA EXAMINADORA

Documento assinado eletronicamente por:

- **Janderson Jason Barbosa Aguiar** (***.765.854-**), em **26/06/2025 23:06:21** com chave **525103e652fb11f0be822618257239a1**.
- **Sabrina de Figueirêdo Souto** (***.047.964-**), em **27/06/2025 08:01:01** com chave **034117c8534611f0a12c06adb0a3afce**.
- **Ana Isabella Muniz Leite** (***.834.864-**), em **26/06/2025 22:47:51** com chave **bc434a8252f811f0aa3406adb0a3afce**.

Documento emitido pelo SUAP. Para comprovar sua autenticidade, faça a leitura do QrCode ao lado ou acesse https://suap.uepb.edu.br/comum/autenticar_documento/ e informe os dados a seguir.
Tipo de Documento: Folha de Aprovação do Projeto Final
Data da Emissão: 27/06/2025
Código de Autenticação: df59ca

À minha mãe e às minhas irmãs, pelo amor e apoio de sempre.

AGRADECIMENTOS

Gostaria de expressar, em primeiro lugar, minha profunda gratidão à minha mãe, uma mulher batalhadora que, com coragem e dedicação, criou a mim e às minhas irmãs, sempre garantindo que nada nos faltasse. Agradeço também às minhas irmãs pelo exemplo inspirador que representam, sendo as primeiras pessoas de nossa família a concluírem o ensino superior. Em especial, agradeço a Emanuely Mabrine, por sempre cuidar de mim com zelo e afeto, como uma verdadeira segunda mãe, e a Evelyne Morgana, pelo brilhante exemplo acadêmico, sendo a primeira mestra da família e futura doutora.

Estendo meus sinceros agradecimentos a todos os familiares que, de alguma forma, contribuíram e me apoiaram ao longo desta jornada.

Agradeço também aos amigos que fiz durante essa caminhada e que estiveram ao meu lado nos momentos bons e, principalmente, nos mais difíceis. Minha gratidão à Joyce Lima, por ser uma companheira excepcional, por seu constante apoio, parceria e incentivo. Agradeço ainda ao meu amigo Caio, cuja amizade construí durante essa trajetória e pelo qual sou imensamente grato.

Por fim, agradeço ao Campus I da Universidade Estadual da Paraíba (UEPB) pela oportunidade de cursar o ensino superior, bem como a todos os professores que contribuíram com seus conhecimentos, dedicação e incentivo ao longo da minha formação, e também a FAPESQ, pela oportunidade do projeto de PIBIC na cota 2023/2024. Em especial, agradeço à professora Ana Isabella Muniz Leite, cuja orientação e apoio tornaram esta experiência ainda mais enriquecedora e significativa.

RESUMO

A segurança é uma área que vem em um crescimento significativo nas últimas décadas, e sua presença em projetos é crucial. No entanto, ao explorar a integração com metodologias ágeis, surge um aspecto de resistência quando se trata da propriedade não funcional da segurança. Essa resistência pode surgir porque o contexto ágil não é propício para algumas práticas de segurança. A metodologia ágil foca na mudança constante e na necessidade constante de satisfazer o cliente. Essas são duas de várias características que entram em conflito com a segurança, pois podem levar a requisitos de segurança mal planejados ou inexistentes. Acidentes recentes mostraram que várias falhas resultaram de interpretações equivocadas dos requisitos de segurança por equipes ágeis. Além disso, ainda há uma carência de soluções que garantam que esses requisitos sejam devidamente tratados tanto na arquitetura do software quanto na implementação. O principal objetivo deste trabalho é apresentar um método (denominado SecurityRE), composto por um metamodelo de acordo com as normas de segurança e diretrizes para especificação de requisitos de segurança, com o intuito de apoiar decisões de design que os considerem em contextos ágeis. Adotamos a abordagem de Design Science Research (DSR) para desenvolver o SecurityRE. A validação do método proposto foi conduzida através de um estudo de caso piloto, com foco na adequação e eficácia do SecutiryRE. Nossos resultados indicam que o SecurityRE é promissor para projetos industriais, especialmente por favorecer a compreensão dos requisitos de segurança pelas equipes ágeis. A comunicação das necessidades relacionadas à segurança para a equipe ágil é crucial para o sucesso de projetos de sistemas críticos à segurança, e o SecurityRE pode servir como um meio para promover essa comunicação e, em última instância, pode contribuir para especificações de segurança de software mais precisas e para facilitar a resolução de conflitos e a tomada de decisões.

Palavras-chave: segurança; sistemas de saúde; ágil; requisitos de segurança;

ABSTRACT

Security has seen significant growth in recent decades, and its presence in projects is crucial. However, when exploring its integration with agile methodologies, resistance often emerges regarding the non-functional property of security. This resistance may arise because the agile context is not always conducive to certain security practices. Agile methodology focuses on constant change and the continuous need to satisfy the revealed that several failures have occurred due to customer. These are two among several characteristics that conflict with security, as they can lead to poorly planned or even absent security requirements. Recent accidents have shown that several failures have resulted from misunderstandings of security requirements by agile teams. Furthermore, there is still a lack of solutions that ensure that these requirements are adequately addressed in both software architecture and implementation. The main goal of this work is to present a method (named SecurityRE), composed of a metamodel aligned with safety standards and guidelines for the specification of safety requirements, aiming to support design decisions that consider them in agile contexts. We adopted the Design Science Research (DSR) approach to develop SecurityRE. We validated our method conducting a pilot case study, with a special focus on assessing the suitability and effectiveness of SecurityRE. Our results indicate that SecurityRE is promising for industrial projects, especially by enhancing the understanding of security requirements by agile teams. Communicating security-related needs to the agile team is crucial for the success of projects, and SecurityRE can serve as a means to promote such communication and ultimately contribute to more accurate software safety specifications and facilitate conflict resolution and decision making.

Keywords: security 1. health system 2. agile 3. security requirement 4.

SUMÁRIO

Página

LISTA DE ILUSTRAÇÕES

1 INTRODUÇÃO

Este trabalho é oriundo de uma pesquisa de Iniciação Científica (PIBIC), com duração aproximada de dois anos, realizada com o apoio da Fundação de Apoio à Pesquisa do Estado da Paraíba (FAPESQ). Como resultado dessa pesquisa, temos desenvolvido o artigo intitulado "Towards a Security Requirement Specification Approach for Health Systems in Agile Contexts", que apresenta os principais achados obtidos ao longo do estudo. Este Trabalho de Conclusão de Curso (TCC) configura-se como uma extensão dessa pesquisa, aprofundando aspectos conceituais e metodológicos e incorporando atividades específicas, como a aplicação do modelo proposto em um estudo de caso controlado, com o objetivo de avaliar sua utilidade, adequação e conformidade frente a normas de segurança da informação.

1.1 Motivação

Segurança é uma área que tem apresentado um crescimento significativo nas últimas décadas, sendo sua presença em projetos crucial. No entanto, ao se explorar a integração com metodologias ágeis, surge um aspecto de resistência quando se trata da propriedade não funcional da segurança, como evidenciado em Nina, Pow-Sang e Villavicencio (2021). Tal resistência pode surgir porque o contexto ágil não é propício a algumas práticas de segurança. A metodologia ágil foca em mudanças constantes e na necessidade constante de satisfazer o cliente. Estas são duas de várias características que entram em conflito com a segurança, pois podem levar a testes de segurança mal planejados ou inexistentes.

Conforme relatado por Goertzel et al. Goertzel et al. (2007), que analisaram as dificuldades de criar software seguro em um ambiente ágil, foi constatado que seis práticas ágeis poderiam ser potencialmente prejudiciais à segurança, três seriam consideradas neutras, e apenas duas seriam positivas. A questão da segurança no desenvolvimento ágil em larga escala (LSAD) apresenta desafios únicos que vão além daqueles encontrados em ambientes ágeis de menor escala.Nägele, Watzelt e Matthes (2022) apontam que um dos principais problemas nesse contexto é a complexidade da comunicação e coordenação entre uma multiplicidade de equipes. Seu estudo mostra que muitas organizações ainda dependem da gestão centralizada da segurança, o que limita a integração efetiva das práticas de segurança no fluxo de trabalho ágil e aumenta as barreiras de comunicação. Em um estudo complementar, os autores analisam mais de perto o tema da gestão de segurança e conformidade no LSAD e identificam problemas como a falta de conscientização sobre segurança, papéis e responsabilidades pouco claros, e a tensão entre a autonomia promovida pelos métodos ágeis e o controle necessário para garantir a conformidade (Nägele; Schenk; Matthes, 2023). Esses achados sugerem que a promoção de um modelo mais distribuído e adaptável, combinado com a melhoria na comunicação entre equipes, é ne-

cessária para permitir procedimentos de segurança eficientes em um ambiente ágil em larga escala (Nägele; Watzelt; Matthes, 2022; Nägele; Schenk; Matthes, 2023).

Em um ambiente ágil, a comunicação ocorre por meio de histórias de usuário, e devido à popularização do Manifesto Ágil, algumas abordagens semelhantes surgiram no que se refere à segurança, como os casos de uso (Alexander, 2003) e casos de abuso (McDermott; Fox, 1999), bem como a história de segurança apresentada por (Asthana et al., 2012; Gharib, 2024).

Essa especialização continua sendo de extrema importância, pois diversos estudos empíricos recentes destacaram as limitações persistentes das abordagens ágeis para lidar com ameaças à segurança. Notavelmente, as práticas de segurança ainda são amplamente inconsistentes com a natureza iterativa e colaborativa do desenvolvimento ágil. Por exemplo, o trabalho de Ardo, Bass e Gaber (2022a) identificou uma clara lacuna entre profissionais de segurança e profissionais ágeis, especialmente na forma como as atividades de segurança são estruturadas e comunicadas nos papéis e etapas do desenvolvimento.

Uma das principais conclusões desses estudos é que muitas práticas de segurança estão focadas nas fases de implementação e validação do ciclo de vida de desenvolvimento de software (SDLC), enquanto aquelas das fases de requisitos e design são frequentemente negligenciadas (Ardo; Bass; Gaber, 2022a). Isso reflete o problema mais amplo da integração insuficiente da segurança desde as fases iniciais do desenvolvimento, o que é essencial para sistemas seguros.

Outro problema importante relatado é a ausência de cerimônias colaborativas e de papéis estruturados para garantir responsabilidade nas equipes ágeis. Em resposta, alguns autores sugeriram procedimentos e papéis específicos, como *sprints* de segurança, cerimônias de conformidade e *backlogs* de segurança dedicados, para melhorar a integração (Ardo; Bass; Gaber, 2022b). A introdução de papéis como especialistas em segurança e testadores de penetração, bem como o uso de ferramentas e checklists de auditoria e conformidade, foram propostos como formas de reduzir a desconexão entre equipes ágeis e de segurança.

Além disso, esses estudos destacam a necessidade de promover uma cultura de comunicação e vigilância constante. Sem um entendimento comum das responsabilidades de segurança e trocas regulares de conhecimento entre desenvolvedores, testadores e profissionais de segurança, as equipes ágeis correm o risco de cair em silos onde o conhecimento é monopolizado e ameaças críticas são ignoradas (Bezerra; Sampaio; Marinho, 2020). Portanto, permitir uma cooperação e comunicação consistentes, não apenas por meio de documentação, mas também por meio de processos e cerimônias claramente definidos, é crucial para alinhar a agilidade com os objetivos de segurança.

Diversos estudos examinaram os desafios de gerenciar a segurança em ambientes onde metodologias ágeis são utilizadas. O trabalho de Tøndel et al. (Tøndel; Jaatun, 2022) propõe um framework conceitual para apoiar a engenharia de segurança em projetos

ágeis. Embora haja um consenso geral sobre a importância da integração entre segurança e agilidade, as abordagens atuais não atendem plenamente às necessidades que surgem durante a fase de desenvolvimento de software e sistemas. Com base em estudos empíricos em ambientes industriais, o framework destaca a lacuna significativa no entendimento do impacto das práticas de segurança nesses projetos. O estudo aponta que ainda há conhecimento limitado sobre como diferentes abordagens para lidar com requisitos de segurança afetam os resultados em diferentes contextos, uma lacuna que permanece amplamente inexplorada na literatura.

Os autores também enfatizam que a escolha de métodos eficazes depende do entendimento da especificidade de cada projeto, incluindo fatores organizacionais, técnicos e culturais. Deve-se também destacar que, em um ambiente ágil, onde a expertise interdisciplinar é constante, a comunicação desempenha um papel fundamental ao informar os profissionais sobre as melhores práticas a serem seguidas, aumentando assim a eficiência dos esforços de segurança no contexto dos ciclos de desenvolvimento ágil. A transferência de informações entre stakeholders, como desenvolvedores, analistas de requisitos e profissionais de segurança, pode levar a mal-entendidos, lacunas de conhecimento ou até mesmo prioridades conflitantes, o que pode impactar negativamente a implementação dos requisitos de segurança. Estratégias que promovam uma comunicação clara, consistente e sensível ao contexto são, portanto, essenciais para garantir que decisões relacionadas à segurança sejam bem compreendidas e devidamente aplicadas ao longo de todo o ciclo de vida do projeto.

1.2 Objetivos

Este cenário evidencia a crescente adoção de métodos ágeis no processo de desenvolvimento de sistemas, qual não é acompanhada por uma inclusão abrangente das práticas de segurança necessárias para o desenvolvimento de software seguro. Essa integração é essencial, especialmente para softwares que precisam de regulação, como softwares aplicados ao domínio da saúde e automotivo. O principal objetivo deste trabalho é contribuir para o aprimoramento da compreensão dos requisitos de segurança de software pelas equipes ágeis e assim a comunicação sobre os objetivos de segurança do projeto.

1.3 Método

Para isso, apresentamos o SecurityRE, um metamodelo baseado nas normas e regulações de segurança para dar suporte as equipes ágeis na derivação de requisitos de segurança, mais especificamente na definição de estratégias explícitas de segurança para prevenção, detecção e mitigação de ameaças. Adotamos a abordagem de Design Science Research (DSR) (Wieringa, 2014a) para desenvolver e validar o SecurityRE, em cooperação com o NUTES[1] (que possui competência no desenvolvimento de software em-

[1]http://nutes.uepb.edu.br/

barcado, avaliação de usabilidade e análise de conformidade de dispositivos médicos). Valiamos o SecurityRE através da condução de um estudo de caso piloto, cujo objetivo era analisar a eficácia do SecurityRE na especificação de requisitos de segurança. Nossos resultados iniciais mostram que a aplicação do SecurityRE é promissora no suporte para uma melhor compreensão da especificação de requisitos de segurança e à redução do tempo necessário para sua análise.

1.4 Estrutura do trabalho

O restante deste documento está estruturado da seguinte forma: a Seção II discute a fundamentação teórica, a Seção III apresenta a metodologia, enquanto a Seção IV apresenta o processo para derivação de requisitos de segurança. Os resultados preliminares são discutidos na Seção V. Finalmente, a Seção VI conclui com um resumo dos resultados e uma perspectiva sobre trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo fornecer o contexto para este trabalho de pesquisa. Assim, são abordadas as definições, termos e explicações relevantes sobre segurança, desenvolvimento ágil e principalmente requisitos de segurança. A seção 2.1 apresenta uma visão geral sobre segurança da Informação. Em seguida, a seção 2.2 descreve práticas e problemas relacionados à especificação de requisitos de segurança. A seção 2.3 apresenta uma visão geral sobre desenvolvimento ágil. Finalmente, a seção 2.4 discorre sobre os trabalhos diretamente relacionados a este.

2.1 Segurança da Informação

O conceito de segurança da informação tem sido amplamente discutido na literatura, sendo apresentado sob diferentes perspectivas que enfatizam aspectos técnicos, organizacionais e contextuais. Segundo Venter et al. (2014), a segurança da informação envolve a proteção dos dados contra diversas ameaças, com o objetivo de reduzir riscos às operações do negócio, aumentar o retorno sobre o investimento, aproveitar oportunidades comerciais e garantir a continuidade operacional. Para isso, os esforços na área devem focar nos princípios fundamentais de integridade, confidencialidade e disponibilidade dos ativos de informação .

De maneira semelhante, Althonayan e Andronache (2018) destaca que a segurança da informação abrange esses mesmos princípios, integridade, confidencialidade e disponibilidade, independentemente do formato dos dados, seja físico ou digital . Essa visão é reforçada pela norma internacional ISO 27002, que define segurança da informação como a proteção da integridade, confidencialidade e disponibilidade da informação em diversos formatos, incluindo digital, impresso, áudio e vídeo (Santos, 2021).

Além disso, Reid e Niekerk (2014) amplia o conceito ao incluir a proteção dos componentes essenciais da informação, como os sistemas e equipamentos que processam, armazenam e transmitem esses dados . Essa perspectiva técnica é complementada por Lundgren, que considera a segurança da informação como a adequação do acesso, levando em conta não apenas propriedades técnicas, mas também aspectos éticos e contextuais. Segundo Lundgren e Möller (2019), esse enfoque permite uma análise aprofundada dos riscos e conflitos de valores entre diferentes stakeholders, o que é fundamental para evitar falhas e vulnerabilidades.

Assim, é possível observar que a segurança da informação é um campo multidimensional, que pode transpassar a simples proteção técnica e partir a incluir aspectos organizacionais, estratégicos e éticos, sendo guiada pelos pilares essenciais da integridade, confidencialidade e disponibilidade.

2.2 Requisitos de Segurança

Os requisitos de segurança representam uma parte essencial do desenvolvimento de sistemas, pois estabelecem as condições necessárias para garantir a proteção dos ativos de informação. Segundo Pfleeger (1991), esses requisitos podem ser entendidos como requisitos funcionais ou de desempenho impostos a um sistema com o objetivo de assegurar um determinado nível de proteção . Já de acordo com Salini e Kanmani (2012), os requisitos de segurança podem ser especificados como restrições à funcionalidade do sistema, sendo essas restrições projetadas para atender a um ou mais objetivos de segurança, como confidencialidade, integridade e disponibilidade. Destaca que a segurança não é apenas uma funcionalidade adicional, mas uma característica que impõe limites e condições específicas ao comportamento esperado do sistema. Uma definição consolidada pode ser encontrada nos documentos do Common Criteria (CC99), que descrevem os requisitos de segurança como a tradução dos objetivos de segurança de TI em especificações técnicas voltadas a garantir a proteção e a confiabilidade do sistema e de seu ambiente (Mohammad; Nazir; Mustafa, 2019; Common Criteria Recognition Arrangement, 2022), enfatizando a importância da formalização e padronização como mecanismos para garantir que os objetivos de segurança sejam de fato implementados.

Dessa forma, os requisitos de segurança atuam como uma ponte entre os objetivos abstratos de proteção e as implementações técnicas concretas, sendo fundamentais para garantir a resiliência do sistema frente a ameaças internas e externas.

2.3 Desenvolvimento Ágil

Os métodos ágeis se destacam por sua ênfase na satisfação do cliente, realizada através de entregas pontuais e regulares de uma versão do software funcional, geralmente essa entrega é realizada por iterações curtas. Esse tipo de desenvolvimento incremental fornece a liberdade de mudanças flexíveis nos requisitos do sistema, além de buscar envolver clientes e partes interessadas do negócio diretamente no processo de desenvolvimento.(Goertzel et al., 2007).

Além disso, as abordagens ágeis são de natureza cíclica, o que a diferencia significativamente do processo tradicional, conhecido como modelo cascata, que segmenta a coleta de requisitos, design, implementação e testes em etapas lineares. Essa característica iterativa dos métodos ágeis busca permitir um ciclo contínuo de feedbacks e melhoria, facilitando a adaptação às mudanças e a entrega contínua de valor (Beznosov; Kruchten, 2005). Já de acordo com Sommerville (2016), o desenvolvimento ágil surgiu como um meio de resposta à rigidez dos modelos tradicionais, priorizando práticas que promovem a entrega contínua de valor ao cliente e a adaptação rápida às mudanças. Os processos ágeis favorecem equipes autogerenciadas, comunicação frequente e desenvolvimento iterativo e incremental, aspectos que contribuem para uma maior responsividade diante de novos requisitos ou

alterações no escopo do projeto .

Assim, o desenvolvimento ágil representa uma transformação nos processos de engenharia de software, privilegiando a flexibilidade, a colaboração e a entrega rápida, o que pode impactar diretamente a maneira como requisitos de segurança são incorporados durante o desenvolvimento.

2.4 Trabalhos Relacionados

Com relação aos trabalhos relacionados, nós conduzimos um mapeamento sistemático (seguindo os guidelines propostos por (Petersen; Vakkalanka; Kuzniarz, 2015), cujo objetivo é investigar as abordagens existentes para especificar requisitos de segurança. Todo o detalhamento desse mapeamento é apresentado no Apêndice A.

Alexander (2003) apresenta a definição de casos de uso indevido (misuse cases), que são essencialmente casos de uso sob a perspectiva de um agente hostil. São basicamente estruturas textuais que podem utilizar diagramas para melhor compreensão. Seu objetivo é ajudar engenheiros a descobrir e organizar requisitos.

McDermott e Fox (1999) definem um método chamado Abuse Case, que também utiliza casos de uso como base. Consiste em definir uma interação completa entre um ator e o sistema que resulta em dano aos recursos associados a um dos atores, partes interessadas ou até mesmo ao próprio sistema. A principal diferença entre os casos de abuso e o anteriormente mencionado caso de uso indevido é que, ao descrever um abuso, deve-se levar em consideração os recursos, capacidades e objetivos do agente malicioso. Também é possível encontrar métodos que aprimoram ou estendem os anteriormente citados.

Røstad (2006) é responsável por apresentar um método que adiciona notações aos casos de abuso, permitindo assim visualizar vulnerabilidades e ameaças internas, considerando que a notação original não levava esses fatores em conta.

Kordy, Pièvre-Cambacédès e Schweitzer (2014) apresentam o estado atual das abordagens de ataque e defesa baseadas em Grafos Diretos Assimétricos (DAGs). Trabalhos recentes de Schiele e Gadyatskaya (2021) exploram a transformação de árvores de ataque em grafos de ataque, propondo um novo método que contrasta com abordagens anteriores, geralmente focadas na direção oposta. Seu framework de transformação foi avaliado por meio de um estudo de caso, demonstrando sua aplicabilidade. O estudo também aponta oportunidades de expansão futura, como a extensão da transformação para Árvores de Ataque e Defesa (AD Trees) e a melhoria da eficiência do algoritmo de transformação em termos de criação de estados, com o objetivo de otimizar tanto o espaço quanto a complexidade temporal. Compreender a evolução das técnicas gráficas de modelagem de segurança é essencial para valorizar plenamente tais desenvolvimentos. Os DAGs, por exemplo, surgiram a partir das "árvores lógicas de ameaça" introduzidas por Weiss (1991) como a primeira técnica gráfica para modelar ataques. Além disso, há várias derivações e

implementações dos DAGs, que são discutidas de forma abrangente no trabalho de Kordy, Pièvre-Cambacédès e Schweitzer (2014).

Abordagens recentes para melhorar a segurança durante o processo de engenharia de requisitos propuseram novos frameworks mais acessíveis. Subha e Haldorai (2022) introduz o método ISTDRE, uma abordagem simples e repetível para identificar ameaças de segurança na fase de design, com o objetivo de simplificar a análise de segurança. Enquanto isso, Kotenko et al. (2023) investigou o perfilamento de atacantes para análise de risco utilizando aprendizado de máquina. Ao analisar o histórico do bash em competições de hacking, demonstrou-se que as equipes invasoras eram distinguíveis por seu comportamento e obtiveram resultados notáveis de classificação.

Janisar et al. (2023) apresentam um framework para apoiar o processo de engenharia de requisitos de segurança por meio do uso de casos de garantia durante a fase de elicitação de requisitos. Essa estrutura consiste nas fases: (1) identificação de ativos, (2) detecção de ameaças, (3) definição de objetivos de segurança, (4) identificação de requisitos de segurança e (5) avaliação dos requisitos de segurança. No contexto da adaptação de procedimentos de segurança ao desenvolvimento ágil de software, Mihelič, Vrhovec e Hovelja (2023) propôs uma abordagem leve para a avaliação e integração de características de segurança em metodologias ágeis. Ao categorizar procedimentos de segurança em papéis, atividades e artefatos, e avaliar os trade-offs entre custo, agilidade e segurança, o estudo oferece aos profissionais um método prático para melhorar incrementalmente a segurança sem sobrecarregar os procedimentos existentes.

Além disso, Rasheed et al. (2021) destacou os problemas críticos encontrados no contexto da Engenharia de Requisitos (RE) em um ambiente ágil. Embora a RE ágil ofereça flexibilidade e retorno mais rápido ao cliente, também introduz problemas de qualidade, cronograma e procedimentos pouco claros. Rasheed enfatizou a imaturidade das práticas de RE ágil e a necessidade urgente de orientações padronizadas para melhorar os resultados.

Bernsmed et al. (2022) examinaram a adoção de atividades de modelagem de ameaças em quatro estudos diferentes dentro de um contexto ágil. Eles também discutiram o que poderia ser feito para facilitar esse processo e observaram que introduzir atividades de segurança de software em um ciclo de vida de desenvolvimento ágil não é uma tarefa fácil.

A análise comparativa realizada entre os oito métodos voltados à especificação de requisitos de segurança em sistemas de software evidencia uma certa diversidade de abordagens, técnicas e níveis de consolidação. A Tabela 2.1 sintetiza as características principais de cada um desses métodos, destacando seus fundamentos, pontos em comum e a existência ou não de ferramentas que apoiem sua aplicação prática.

De maneira geral, observa-se que a maioria dos métodos se apoia em notações conhecidas, como a UML (Unified Modeling Language), o que facilita sua compreensão e adoção por parte de desenvolvedores e analistas. Métodos como Abuse Cases, Misuse Cases e

Método	Técnica Principal	Pontos em Comum	Ferramenta
SecureUML	UML + RBAC + OCL	Uso de UML, foco em controle de acesso, integra segurança à modelagem	Não
Abuse Cases	Casos de uso adaptados para ataques	Usa notação de casos de uso; fácil de entender; foco na análise de ameaças	Não
Misuse Cases	Extensão de casos de uso UML	Uso de atores hostis; relações "threatens" e "mitigates"; comunicação com stakeholders	Não
Extended Misuse Cases	Misuse Cases + Insiders + Vulnerabilidades	Estende misuse cases com atores internos e vulnerabilidades; foca em riscos reais e modelagem rica	Não
SQUARE	Processo sistemático de 9 etapas	Elicitação e priorização sistemática de requisitos; integração no início do ciclo de vida	Sim
STORE	Análise de ameaças e pontos de ataque	Foco na elicitação sistemática; orientação por ameaças; define PoA, PoB, PoC, PoD	Não
US4USec	User Stories adaptadas para segurança	Foco em requisitos de segurança usáveis; integração com histórias de usuário (NFR + user-centered)	Não
Secure Tropos	Extensão orientada à segurança do Tropos	Integra modelagem de requisitos sociais e técnicos; foco em análise formal e socio-organizacional	Sim

Tabela 2.1 – Abordagens para especificação de requisitos de segurança. *(Fonte: Elaborado pelo autor)*

Extended Misuse Cases exemplificam essa tendência, utilizando ou adaptando casos de uso UML para expressar comportamentos indesejados, ameaças ou vulnerabilidades. Essa familiaridade contribui para a comunicação efetiva entre especialistas e não-especialistas em segurança.

Embora esses métodos não contem com ferramentas dedicadas ou automatizadas, todos compartilham a capacidade de poderem ser modelados em qualquer ferramenta UML tradicional, como Visual Paradigm, StarUML, Draw.io ou FlowChart. Isso é possível devido à sua aderência à notação de casos de uso e à simplicidade gráfica que os caracteriza. A mesma lógica se aplica ao SecureUML, que, apesar de possuir protótipos específicos para geração de infraestrutura de controle de acesso, também pode ser modelado em qualquer ambiente UML que aceite extensões.

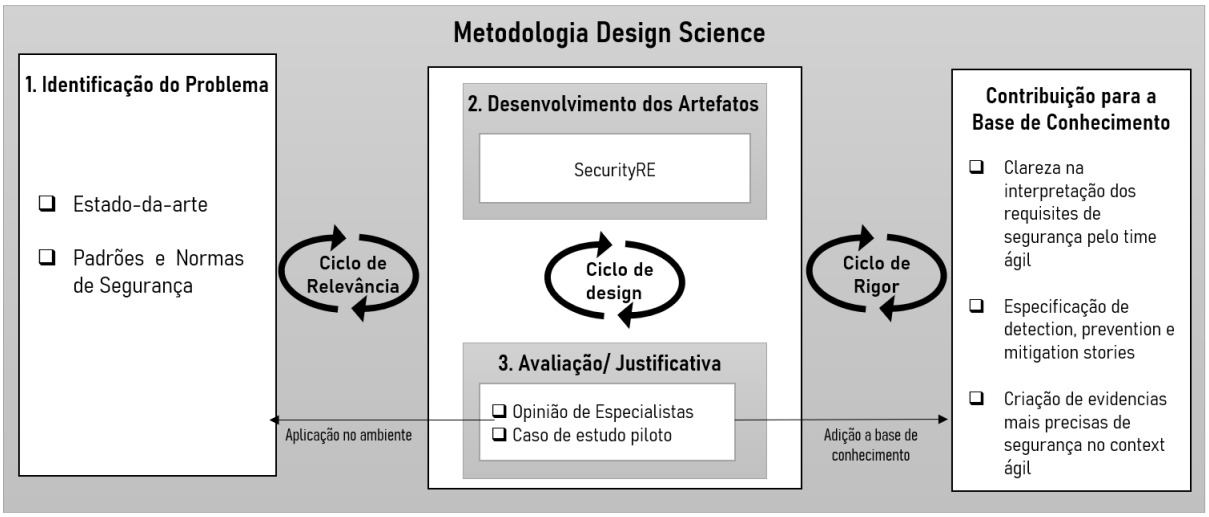
Apesar dessa similaridade quanto ao suporte por ferramentas UML genéricas, existem diferenças conceituais relevantes entre os três métodos. Abuse Cases propõem a representação de cenários de ameaça como casos de uso abusivos, comumente modelados separadamente dos casos de uso funcionais. Misuse Cases integram ameaças e funcionalidades legítimas em um mesmo diagrama, permitindo a visualização de interações entre casos de uso e atores maliciosos. Já o modelo Extended Misuse Case avança ao incorporar a representação explícita de vulnerabilidades e a diferenciação entre atacantes externos e internos, aumentando a expressividade do modelo, mas também a complexidade de sua aplicação prática.

No que se refere ao suporte por ferramentas, identificou-se que o método SQUARE possui a ferramenta eSQUARE(Abukwaik; Zhang, 2012) , que fornece suporte a todas as etapas da metodologia, auxiliando na sistematização e rastreabilidade das atividades realizadas. Partindo para a metodologia Secure Tropos, a mesma se destaca por contar com a ST-Tool, uma ferramenta robusta que oferece um ambiente gráfico para modelagem e integra recursos de verificação automática. Esse suporte de ferramentas torna o método mais aplicável em contextos que exigem maior rigor na verificação e validação dos requisitos de segurança.

3 METODOLOGIA

Para conduzir este trabalho, nós seguimos a metodologia de Design Science (DSR) (Wieringa, 2014b) (como apresentada na Figura 1), através da qual os artefatos em nossa pesquisa são desenvolvidos e validados iterativamente no domínio real.

Figura 1 – Visão geral da metodologia design-science



(Fonte: Adaptada de Wieringa (2014b)).

Identificação do Problema: Esta etapa é caracterizada pela investigação do estado da arte, sob uma perspectiva de pesquisa, revisando e discutindo trabalhos relacionados. Para isso, realizamos um mapeamento sistemático da literatura para fundamentar nosso estudo, buscando sistematicamente a literatura sobre como a segurança tem sido abordada ao longo do desenvolvimento ágil e principalmente as abordagens existentes para a especificação de requisitos de segurança. Além disso, temos analisado as normas de segurança buscando identificar quais informações de segurança devem ser consideradas para especificar medidas de detecção e prevenção de vulnerabilidade de segurança, a fim de levar o sistema a um estado seguro.

Desenvolvimento dos artefatos: Nesta etapa, propomos uma contribuição para suportar a especificação de requisitos de segurança no contexto ágil. Essa contribuição será composta por um metamodelo e diretrizes, mais especificamente: eles possibilitarão a especificação explicita de estratégias de segurança centradas em detectar, prevenir e mitigar vulnerabilidades. Nesta fase, elicitamos e desenvolvemos o metamodelo e as diretrizes. As primeiras ideias para o metamodelo potencial foram extraídas com base em nossas análises das normas de segurança e revisão da literatura. Em seguida, organizamos rodadas de conversas com equipes ágeis e especialistas em segurança para discutir os achados iniciais.

Avaliação dos artefatos: Esta etapa está relacionada à validação e refinamento da nossa contribuição. O desenvolvimento e avaliação dos artefatos foram conduzidos por meio de múltiplos ciclos de design, para que os artefatos fossem estudados e aprimorados iterativamente com o suporte direto dos colaboradores. Pretendemos melhorar a compreensibilidade dos requisitos de segurança pela equipe ágil no que diz respeito às histórias de segurança e à arquitetura de software, em termos de eficácia eficiencia. A eficácia refere-se à especificação de requisitos de segurança mais precisos no contexto ágil. A eficiência refere-se à especificação rápida e precisa desses requisitos (just-enough). Para isso, conduzimos um estudo de caso piloto, de acordo com as diretrizes estabelecidas por Wohlin et al. (2012), Jedlitschka, Ciolkowski e Pfahl (2008), dentro do escopo do projeto da Senior Saúde Móvel[1] desenvolvido em parceria pelo Núcleo de Tecnologias Estratégicas em Saúde (NUTES[2]).O estudo piloto foi conduzido com alunos de graduação do CCT/UEPB, visando validar o protocolo e instrumentos do caso de estudo (conforme detalhado no Apendice B e Apendice C. Como também, para cumprir o ciclo de rigor em nossa metodologia, revisar trabalhos existentes e refletir sobre como nossos resultados de pesquisa são transferíveis para outros contextos.

[1] https://senior.nutes.uepb.edu.br/
[2] ⟨http://nutes.uepb.edu.br/⟩

4 SECURITYRE

Este capítulo apresenta o SecurityRE, que busca melhorar a compreensão dos requisitos de segurança pelas equipes ágeis, destacando métricas de segurança para os desenvolvedores. Com o uso do modelo, as equipes ágeis podem construir o conhecimento compartilhado necessário para lidar adequadamente com preocupações de segurança e fornecer evidências mais precisas e tangíveis aos órgãos certificadores de que o escopo de segurança foi definido e colocado em prática.

O modelo SecurityRE representado na Figura 2, foi desenvolvido para auxiliar equipes ágeis na identificação e processamento de uma gama de informações relevantes do ponto de vista arquitetural, a fim de derivar estratégias para prevenção, mitigação e resposta a vulnerabilidades.

Muitos projetos, especialmente os críticos, exigem conformidade com padrões rigorosos de segurança e qualidade. Por exemplo, projetos na área da saúde frequentemente precisam estar em conformidade com a Lei de Portabilidade e Responsabilidade de Seguro de Saúde (HIPAA) (United States, 1996), a fim de garantir a proteção de dados sensíveis dos pacientes. Além disso, dependendo da natureza do produto, pode ser necessário cumprir normas internacionais, como a ISO ou IEC 81001-5-1 ou a ISO 13485:2016 (International Organization for Standardization, 2021; International Organization for Standardization, 2016).

A norma ISO IEC 81001-5-1 (International Organization for Standardization, 2021) define as atividades de segurança para softwares de saúde e sistemas de informação em saúde ao longo de seu ciclo de vida, enfatizando que essas atividades devem ser realizadas dentro de um sistema de gestão da qualidade documentado, geralmente conforme a ISO 13485 ou equivalente. Isso garante que os procedimentos de segurança sejam sistematicamente estabelecidos, documentados e verificados quanto à sua eficácia. Por sua vez, a ISO 13485:2016 (International Organization for Standardization, 2016), que regula sistemas de gestão da qualidade para dispositivos médicos, destaca a importância da rastreabilidade. A organização deve possuir procedimentos para controlar a identificação, armazenamento, segurança, recuperação e retenção dos registros, de forma a fornecer evidência de conformidade e garantir que as informações permaneçam precisas, seguras e acessíveis durante todo o ciclo de vida do produto. Juntas, essas normas fortalecem a postura de segurança e a prontidão regulatória de sistemas críticos, promovendo maior confiança e confiabilidade em ambientes onde segurança e conformidade são de suma importância.

No entanto, devido à dificuldade inerente ao uso de métodos voltados à segurança em ambientes ágeis, essas medidas muitas vezes são negligenciadas. Há poucas publicações sobre conformidade de segurança no desenvolvimento ágil de software. A maioria das contribuições se concentra em discussões teóricas, com pouca evidência empírica sobre a

integração de requisitos de normas de segurança em metodologias ágeis, em vez de fornecer estratégias específicas para avaliação da conformidade (Moyon et al., 2020).

Nosso modelo oferece orientações essenciais para a especificação de soluções com foco em segurança e permite que a equipe ágil identifique as informações necessárias para o desenvolvimento de security stories e outros artefatos que ajudam a ampliar a compreensão da equipe de desenvolvimento sobre as métricas de segurança do projeto.

O primeiro artefato no desenvolvimento do projeto é o security epic. Sua criação exige uma avaliação das ameaças e vulnerabilidades que podem surgir no projeto. Para cada ameaça, deve-se identificar um security epic e atribuir um nível de risco (Glinz; Wieringa, 2007). No nível de contexto, os security epics definem o contexto de desenvolvimento e agrupam os requisitos associados.

Após a criação dos security epics, nosso modelo avança para outro nível: o refinamento de cada security epic em security stories, representando o nível funcional. Essas security stories são responsáveis por documentar as estratégias de implementação das medidas de segurança estabelecidas. Essas medidas tratam da prevenção, detecção e mitigação de vulnerabilidades (Valdés-Rodríguez et al., 2023; Reznik, 2022), considerando o risco associado a cada security epic. Portanto, cada *security story* deve ser refinada em uma *prevention story*, *detection story* e *mitigation story*.

As prevention, detection e mitigation stories podem incluir estratégias que nem sempre exigem implementação via software. No entanto, em nosso modelo, é obrigatória uma implementação em nível de software para essas histórias. Assim, toda prevention, detection e *mitigation story* deve corresponder a uma estratégia de prevenção, uma técnica de detecção e uma estratégia de mitigação, respectivamente. Essas estratégias e técnicas são implementadas por meio de componentes, soluções e/ou interfaces externas.

As estratégias de prevenção devem ser implementadas com componentes de segurança reutilizáveis, ou seja, componentes de segurança já consolidados no mercado, oriundos de fontes confiáveis e com suporte de atualização robusto, como OAuth, TLS, JWT, WAFs, entre outros.

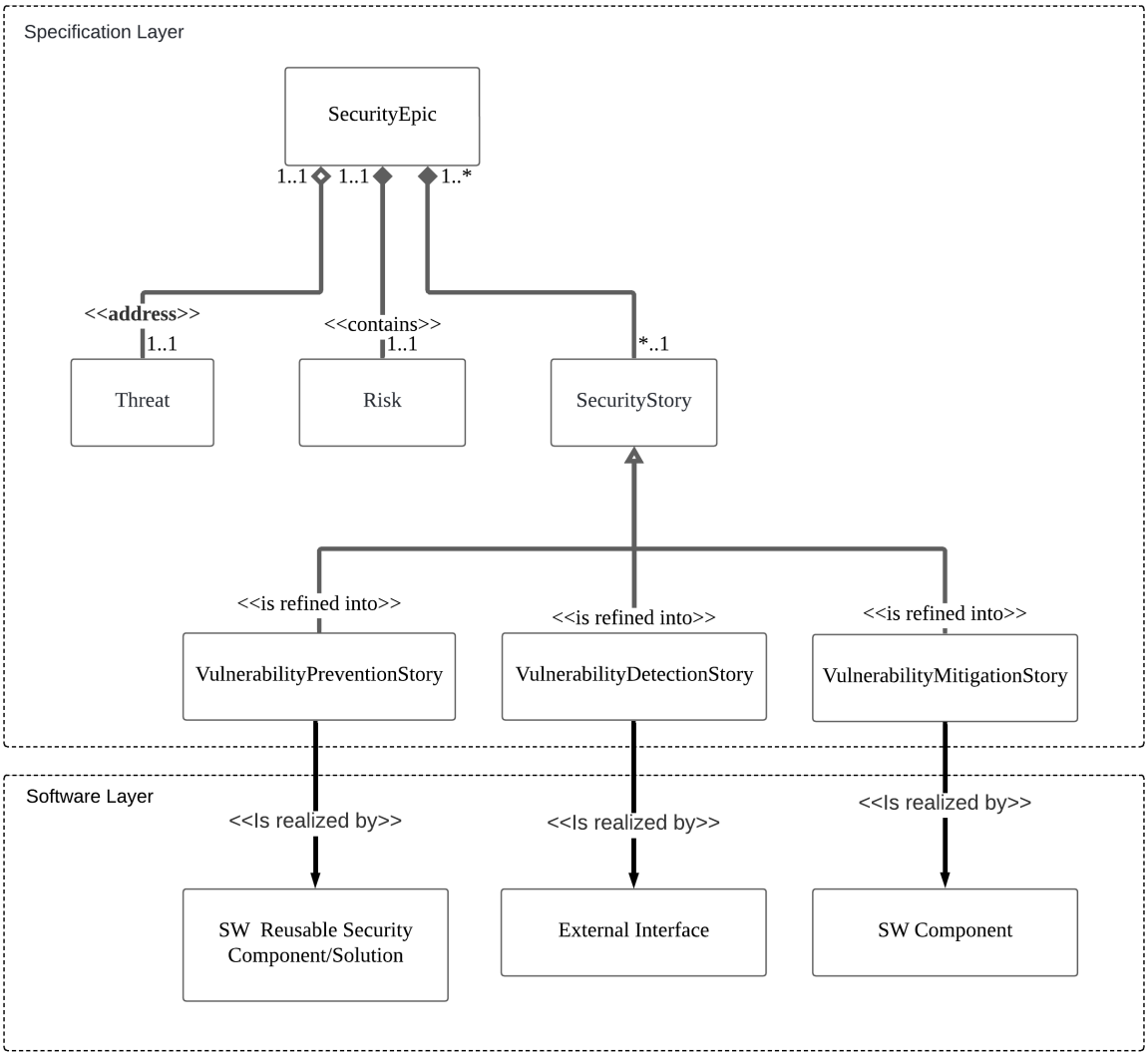
Com relação à *detection story*, as técnicas associadas têm como objetivo identificar lacunas potenciais que não foram tratadas na história de prevenção. Para isso, podem ser definidas múltiplas técnicas de detecção, que devem ser implementadas com o objetivo de maximizar a detecção de vulnerabilidades. Essas técnicas devem ser executadas por meio de interfaces externas, como ferramentas de análise dinâmica de código, análise de composição de software, varredura de vulnerabilidades, registro de logs e monitoramento de segurança, entre outras.

Por fim, temos a *mitigation story*, que visa responder a uma vulnerabilidade específica, ou instância dessa vulnerabilidade, que não foi prevenida na história de prevenção e foi descoberta posteriormente na história de detecção. A *mitigation story* complementa a história de prevenção, pois abrange a fase de remediação da vulnerabilidade identificada.

Essa remediação deve ser realizada por meio do desenvolvimento de um novo componente de software específico, que implemente as medidas para mitigar a vulnerabilidade em questão.

É importante enfatizar que a definição de qualquer prevention, detection e *mitigation story* é sequencial: primeiro prevenir, depois detectar e, se necessário, responder. A definição das estratégias a serem aplicadas em cada etapa é fundamental para evitar correções de grande escala, que podem aumentar significativamente os custos de desenvolvimento. Quanto mais cedo a medida de segurança for implementada, mais eficaz ela tende a ser percebida (Rindell et al., 2021).

Figura 2 – Modelo do processo de derivação



Fonte: Elaborado pelo autor, 2024

4.1 Exemplo de Aplicação em Projeto Real

Neste artigo, utilizamos a plataforma Senior Mobile Health (SMH) de Rodrigues et al. (2022) para validar e exemplificar o Processo de Derivação de Requisitos de Segurança. Os principais objetivos da plataforma são utilizar dados de variabilidade da frequência cardíaca de idosos, obtidos por meio de monitoramento remoto, para melhorar a qualidade de vida e aprimorar a avaliação clínica através da aquisição diária de dados. A plataforma SMH utiliza uma arquitetura baseada em microserviços e Internet das Coisas (IoT). Essas características, juntamente com o fato de a plataforma lidar com dados de saúde, formam um ambiente em que a segurança é um fator crítico. Nove componentes principais compõem a arquitetura da plataforma SMH:

- Express Gateway: o único componente exposto à rede externa;
- Agente de Sincronização de Dados: o serviço responsável por coordenar os dados entre dispositivos vestíveis e a plataforma SMH;
- Canal de Mensagens RabbitMQ: o componente responsável pela comunicação assíncrona entre microserviços;
- Série Temporal: responsável por armazenar informações sobre passos dados, frequência cardíaca, calorias queimadas e distância percorrida;
- Conta: microserviço responsável pela autenticação e armazenamento de informações gerais dos usuários;
- Medição de Dados de Saúde: microserviço responsável por rastrear várias métricas, incluindo pressão arterial, peso, HRV, atividade física e sono;
- Notificação: microserviço encarregado de enviar notificações e mensagens SMS e emails;
- Agente de Notificação: microserviço responsável por armazenar questionários e centralizar regras de negócios;
- Analytics: microserviço encarregado de processar, analisar e fornecer informações sobre a saúde dos idosos.

No primeiro passo, foi necessário compreender a arquitetura da aplicação para derivar requisitos de segurança. Portanto, identificar potenciais vulnerabilidades ou ameaças relacionadas aos microserviços foi imperativo. Hannousse e Yahiouche (2021) apresentam um estudo de mapeamento metódico que nos fornece uma base sólida sobre possíveis ameaças e vulnerabilidades que podem surgir em uma arquitetura de microserviços. A lista de ameaças do artigo e outras listas de ameaças

(Humayun et al., 2020; Baker; Nguyen, 2019) serviram como base para vincular ameaças potenciais aos locais onde poderiam ocorrer, levando em consideração as operações de cada microserviço. No nosso exemplo, utilizamos a ameaça de injeção de SQL, que foi determinada através da análise de sistemas como uma possível ameaça a alguns microserviços da plataforma, como os serviços de Conta e Medição de Dados de Saúde.

O próximo passo é definir as epic de segurança. Diretrizes e normas são utilizadas para esse propósito, como os padrões de segurança da Lei de Portabilidade e Responsabilidade de Seguros de Saúde (HIPAA). Em seguida, avaliamos cada épica e atribuímos um valor com base em sua importância e no feedback das partes interessadas. Com base na análise mencionada, constatou-se que há um alto risco associado à prevenção de injeção de SQL, ou seja, a ameaça de injeção de SQL. Posteriormente, refinamos essa épica em uma história; no nosso exemplo, evidenciado na figura 3, a história de segurança relacionada à prevenção de injeção de SQL é: " Implementar validação de dados de entrada em todos os formulários e parâmetros de URL para evitar injeção de SQL". Precisamos refinar a história de segurança posteriormente, portanto, essa é outra etapa a ser completada. A história de segurança mencionada é dividida em três partes:

1. História de prevenção de vulnerabilidade: aplicando práticas de codificação segura, o desenvolvedor deve estudar e aplicar práticas para prevenir a injeção de SQL.

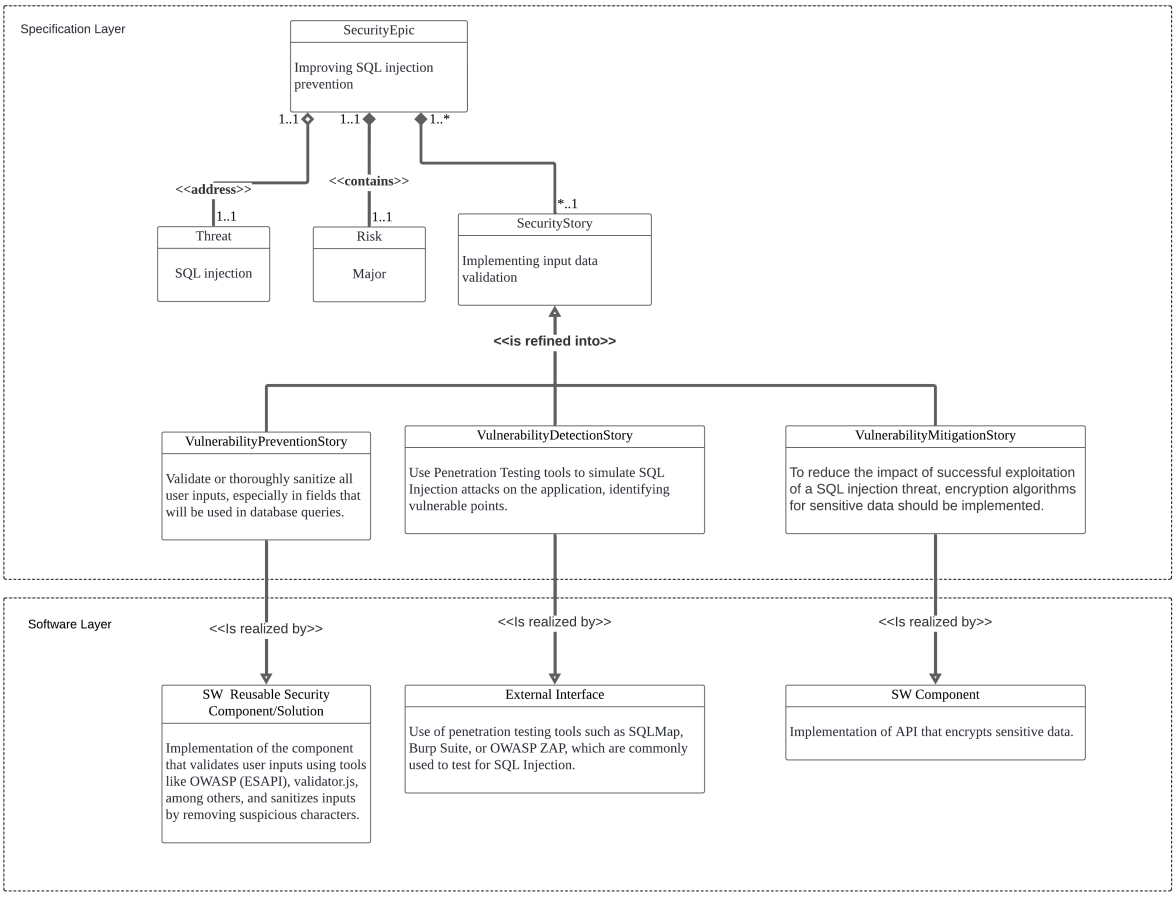
2. História de detecção de vulnerabilidade: utilizando ferramentas internas de testes de penetração, a equipe de segurança ou a parte responsável pelo teste deve usar ferramentas de verificação de injeção de SQL, como ferramentas de análise de código, ou deve haver um especialista capaz de realizar e identificar a vulnerabilidade.

3. História de mitigação de vulnerabilidade: a equipe ou parte responsável deve revisar os resultados do relatório anterior e aplicar ou ajustar os mecanismos de segurança adequados para resolver a vulnerabilidade, o que inclui o criptografia dos dados sensíveis da aplicação.

Subsequentemente, cada história de prevenção, detecção e mitigação deve abordar uma solução de segurança reutilizável para prevenção, uma interface externa para detecção e um componente de software para mitigação. No nosso caso, a solução de segurança escolhida para implementação na história de prevenção, demonstrada na figura 4 foi o Express Validator, uma ferramenta para limpar e validar entradas de dados no sistema. Essa ferramenta pode ser implementada diretamente no microserviço Express Gateway, pois é o único componente da aplicação exposto a interfaces externas e é responsável por validar e sanitizar todos os dados que entram.

Para a história de detecção que podemos encontrar na figura 6, escolhemos o SonarQube, uma plataforma popular para análise contínua de qualidade e segurança de código com suporte para várias linguagens. Ele é capaz de detectar vulnerabilidades,

Figura 3 – Exemplo em execução de ameaça de injeção de SQL

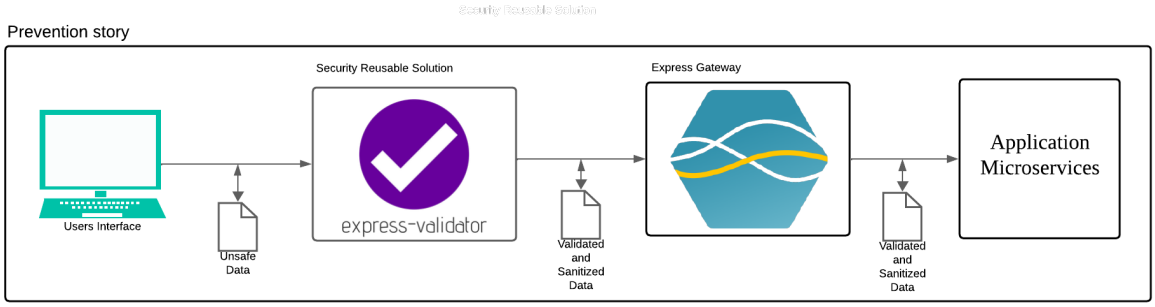


Fonte: Elaborado pelo autor, 2024

integrar-se a pipelines de CI/CD e oferece suporte para certos plugins, como SonarJS e SonarPython. O SonarQube deve ser implementado de forma a monitorar todos os microserviços da aplicação para uma verificação todo o código.

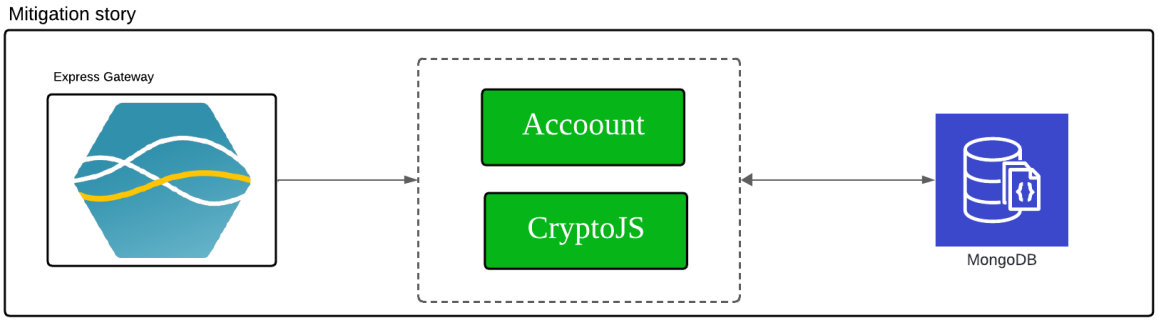
A história de mitigação da figura5 visa reduzir os danos causados pela possível exploração de uma ameaça. No nosso caso, essa ameaça seria a injeção de SQL. No nosso sistema, a criptografia de dados foi selecionada como uma das técnicas de mitigação de riscos. Essa técnica foi escolhida com base no framework MITRE ATT&CK, uma base de conhecimento acessível globalmente sobre táticas e técnicas de adversários, baseada em observações do mundo real. A tecnologia escolhida para implementar a criptografia de dados foi o Crypto.js, um módulo do Node.js que fornece funções de criptografia e hashing diretamente no código. A criptografia seria aplicada a alguns microsserviços, como o Account e o MHealth. Nesses casos, a criptografia poderia ser diretamente no componente, ficando responsável por criptografar os dados e enviá-los ao banco de dados.

Figura 4 – Modelagem da história de prevenção



Fonte: Elaborado pelo autor, 2024

Figura 5 – Modelagem da história de mitigação



Fonte: Elaborado pelo autor, 2024

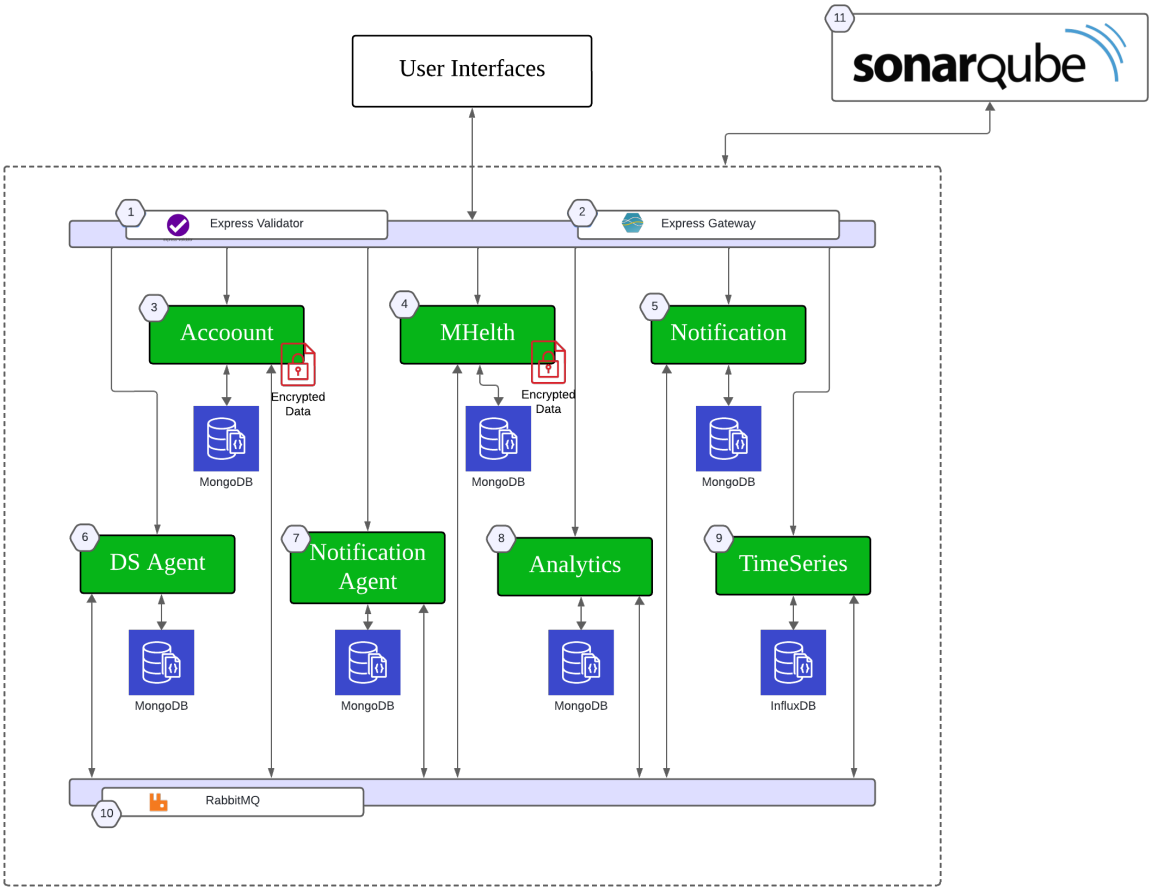
4.2 Estudo de Caso Piloto

Esta etapa está relacionada à avalidação e refinamento da nossa contribuição, como mencionado na seção 3. Para isso, conduzimos um estudo de caso piloto, de acordo com as diretrizes estabelecidas por Wohlin et al. (2012), Jedlitschka, Ciolkowski e Pfahl (2008), dentro do escopo do projeto da Senior Saúde Móvel[1] desenvolvido em parceria pelo Núcleo de Tecnologias Estratégicas em Saúde (NUTES[2]).O estudo piloto foi conduzido com alunos de graduação do CCT/UEPB, visando validar o protocolo e instrumentos do caso de estudo (conforme detalhado no Apendice B e Apendice C.

O principal objetivo do estudo foi analisar a adequação do método à realidade do desenvolvimento ágil, sua conformidade com normas de segurança reconhecidas e sua utilidade para desenvolvedores com pouca ou nenhuma experiência prévia em segurança. A primeira aplicação do estudo foi realizada com a participação de 23 estudantes da área de computação, inseridos em um contexto simulado de desenvolvimento ágil de sistemas de saúde. Está prevista uma segunda rodada da aplicação do estudo, desta vez com

[1] https://senior.nutes.uepb.edu.br/
[2] ⟨http://nutes.uepb.edu.br/⟩

Figura 6 – Arquitetura modificada com a solução de detecção



Fonte: Elaborado pelo autor, 2024

profissionais da área, a fim de obter dados adicionais que permitam uma avaliação mais robusta e comparativa.

A condução do estudo de caso seguiu sete etapas principais: introdução ao projeto e ao método SecurityRE, coleta de consentimento informado dos participantes, aplicação de um questionário inicial para levantamento do perfil dos sujeitos, explicação do método, realização de um treinamento introdutório, execução de uma tarefa prática com aplicação do método e, por fim, aplicação de um questionário pós-atividade com perguntas voltadas ao feedback sobre a experiência. Durante a execução, os participantes foram orientados a aplicar o SecurityRE na identificação e especificação de requisitos de segurança com base em ameaças previamente definidas. O método foi aplicado em um cenário baseado em um sistema real da área da saúde, com foco em ameaças típicas como a injeção de SQL.

A coleta de dados foi realizada por meio de dois instrumentos principais: análise documental dos artefatos produzidos durante a atividade prática e questionários estruturados baseados na escala de Likert. Os questionários foram elaborados para mensurar

três variáveis dependentes: a adequação do método ao contexto ágil, a conformidade dos artefatos gerados com normas de segurança como HIPAA, GDPR e ISO/IEC 27001, e a utilidade do método segundo a percepção dos participantes. Questões como "O método de derivação de requisitos de segurança é claro para você, mesmo tendo pouca ou nenhuma conexão com segurança" e "As histórias de prevenção fornecem diretrizes práticas e aplicáveis para você" foram utilizadas para captar essas percepções.

A análise quantitativa dos dados obtidos nos questionários foi realizada por meio de estatísticas descritivas, buscando identificar padrões e correlações nas respostas. Já a análise qualitativa dos documentos gerados pelos participantes focou na avaliação da completude e conformidade dos requisitos com as normas estabelecidas. Essa verificação foi realizada manualmente, por meio da comparação direta entre os artefatos produzidos pelos estudantes e os principais requisitos presentes em normas como HIPAA, GDPR e ISO/IEC 27001, considerando aspectos como controle de acesso, confidencialidade dos dados e uso de técnicas de criptografia.

4.2.1 Resultados

Os dados obtidos por meio do questionário de avaliação (conforme Apendice D e Apendice E) do método SecurityRE apontam que os participantes, em sua maioria, perceberam o método como claro e compreensível, como podemos ver na figura 7. As respostas sugerem que o processo de derivação de requisitos de segurança foi, em geral, considerado acessível mesmo por aqueles com pouca familiaridade com o tema. Embora alguns participantes tenham apontado dificuldades relacionadas à simplicidade e à execução do método, a percepção geral indica que a proposta foi compreendida durante a atividade prática.

A documentação fornecida foi avaliada positivamente, sendo considerada útil como suporte à realização das tarefas propostas. Também foi identificado que parte dos participantes percebeu que o método contribuiu para melhorar sua capacidade de documentar requisitos de segurança de forma mais clara. No entanto, quanto à redução da complexidade na implementação desses requisitos, as respostas mostraram opiniões variadas, refletindo diferentes níveis de familiaridade prévia com segurança da informação.

No que diz respeito à adequação do método ao desenvolvimento ágil, como podemos ver na figura 8 os dados sugerem uma boa aceitação entre os participantes. As respostas indicam que o SecurityRE foi visto como compatível com práticas comuns do ágil, como sprints e reuniões diárias, e que seu uso pode facilitar a comunicação entre membros da equipe, especialmente em relação aos requisitos de segurança.

Com relação ao perfil dos participantes, verificou-se que todos eram estudantes de graduação, com pouca ou nenhuma experiência prática em segurança da informação. Muitos deles também demonstraram familiaridade limitada com normas como HIPAA, GDPR e ISO/IEC 27001. Apesar disso, os dados sugerem que o método foi compreendido

e considerado útil mesmo por esse público, o que pode indicar uma boa aplicabilidade do SecurityRE em equipes com baixa maturidade em segurança.

Em relação à análise dos documentos gerados durante a atividade prática, foi possível perceber que boa parte dos estudantes não conseguiu produzir os artefatos em sua completude. Esse fator pode estar associado ao tempo limitado disponível para a execução da tarefa, o que possivelmente comprometeu a elaboração total das histórias de prevenção, detecção e mitigação. Diante disso, não foi possível chegar a conclusões concretas quanto à métrica de conformidade (compliance) com as normas de segurança. Esse obstáculo também reforça a necessidade de reaplicação do modelo em um novo ciclo, com ajustes no tempo de aplicação e com a inclusão de participantes mais experientes, a fim de ampliar a compreensão e a solidez dos dados obtidos.

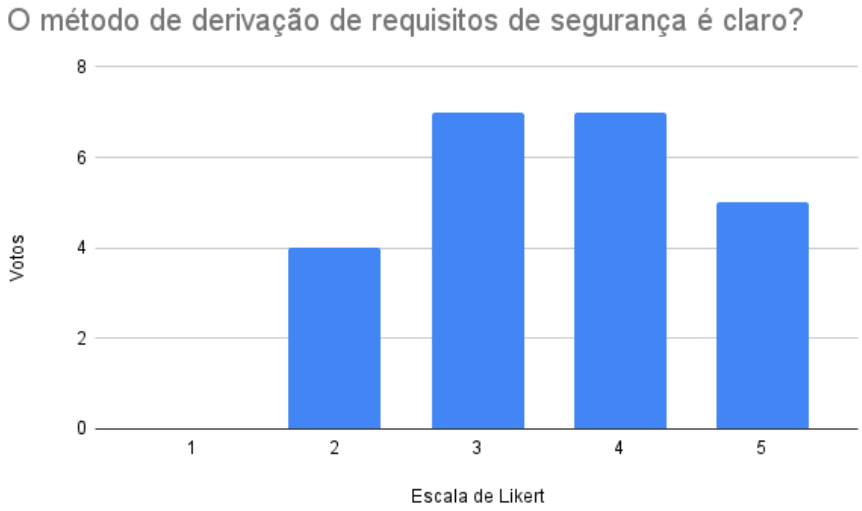


Figura 7 – Respostas sobre clareza do método

Fonte: Elaborado pelo autor, 2024

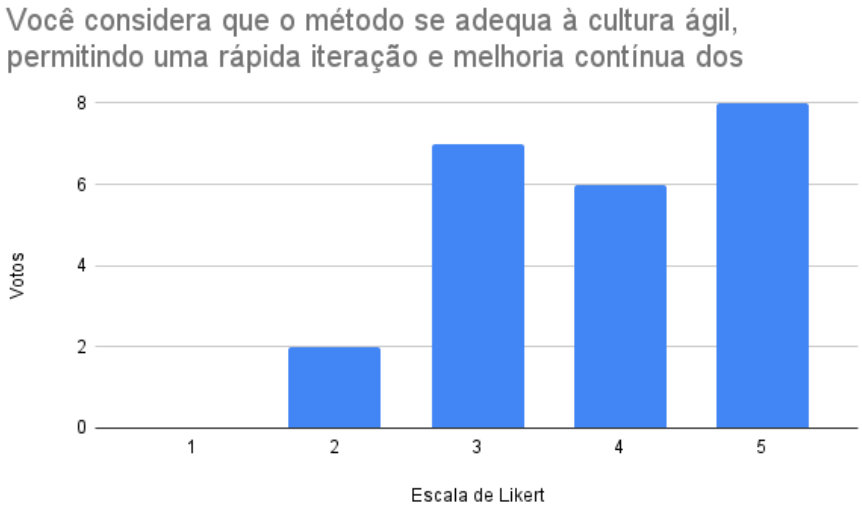


Figura 8 – Adequação com metodologia Ágil

Fonte: Elaborado pelo autor, 2024

5 DISCUSSÃO

Com a crescente popularidade e aplicação das metodologias ágeis, torna-se necessário um melhor entendimento dos requisitos de segurança e uma integração aprimorada de estratégias de segurança, a fim de viabilizar o desenvolvimento de software seguro. No entanto, é evidente que aplicar métodos orientados à segurança em conjunto com métodos ágeis é um processo complexo. Essa complexidade decorre da natureza, por vezes contraditória, entre os dois métodos, bem como da falta de desenvolvedores com formação ou experiência em engenharia de segurança, o que dificulta a compreensão dos requisitos de segurança. O processo de derivação de requisitos descrito aqui tem como objetivo orientar como os processos de segurança devem ser conduzidos, de modo a garantir que as soluções arquiteturais com foco em segurança abordem o maior número possível de preocupações relacionadas à segurança. O estudo de caso piloto mostrou que o método SecurityRE foi, em geral, bem recebido por estudantes com pouca experiência em segurança, especialmente no que diz respeito à clareza e utilidade das diretrizes apresentadas. Os dados apontam ainda que o método é percebido como compatível com práticas ágeis e como um facilitador da comunicação entre membros da equipe.

Entretanto, a análise dos documentos gerados indicou que muitos participantes não conseguiram concluir todos os artefatos propostos, o que comprometeu a avaliação da conformidade com normas como HIPAA, GDPR e ISO/IEC 27001. Isso pode estar relacionado ao tempo reduzido para a atividade, sugerindo a necessidade de ajustes no planejamento de aplicação e a reaplicação do modelo em novos contextos. Essa limitação também reforça a importância de validar o modelo com participantes mais experientes, a fim de obter dados mais robustos e conclusivos.

O uso do modelo descrito neste trabalho visa melhorar a forma como as equipes ágeis lidam com requisitos e métricas de segurança. Consequentemente, esse processo leva a soluções arquiteturais mais precisas e eficientes. Essas soluções também ajudam as equipes ágeis a otimizar a comunicação interna, promover uma visão unificada da criticidade do sistema e facilitar uma interpretação precisa dos requisitos de segurança.

É importante observar que as soluções aqui descritas não abrangem a estrutura detalhada de todos os componentes do sistema. Em vez disso, o modelo busca utilizar informações relevantes para desenvolver estratégias de prevenção, detecção e mitigação. Essas soluções podem ser refinadas ao longo das sprints ágeis. Outro ponto a destacar é que o modelo apresentado não define nenhuma terminologia específica a ser seguida; ele apenas enfatiza que a criação dos artefatos deve ser clara e relevante.

5.1 Ameaças da Validade

A primeira ameaça, à validade externa, refere-se à veracidade aproximada das conclusões que envolvem generalizações em diferentes contextos. Conduzimos nosso trabalho com 23 participantes com diferentes perfis. Como nossa metodologia de design science consiste em várias fases, os participantes estiveram envolvidos em diferentes partes da coleta de dados. A validação piloto do método com estudantes, ajuda a validar o protocolo e os instrumentos utilizados na condução do estudo de caso, o que será fundamental para mitigar ameaças à generalização. Esperamos que nossas descobertas possam ser transferidas para outros domínios, especialmente em contextos ágeis de grande escala.

A segunda ameaça, à validade interna, está relacionada ao ruído nos dados descoberto durante os ciclos iterativos de avaliação da nossa metodologia de design science. Para mitigar vieses e pré-concepções desenvolvidas durante o estudo, discutimos criticamente nossas descobertas com outros pesquisadores e profissionais na área de segurança. Também analisamos cuidadosamente nossos resultados iniciais usando os métodos descritos na Seção 3.

A terceira ameaça, à validade de construção, refere-se ao estabelecimento de medidas operacionais corretas para a construção da solução. Nesse sentido, a efetividade do método foi validada através da condução de um estudo de caso piloto, neste a realização de tarefas de especificação pelos participantes, tiveram seu desempenho operacionalizado pelo tempo gasto (esforço) e pela precisão das soluções. Se um participante compreende a real intenção dos requisitos de segurança, então ele deve ter um desempenho melhor em uma tarefa de especificação e/ou a solução deve ser mais correta. E, se o participante tiver uma melhor compreensão dos requisitos e da tarefa a ser realizada, o tempo gasto deverá ser menor. A análise dos resultados mostrou que todos os participantes foram capazes de fornecer especificações de segurança de software mais precisas.

A quarta ameaça, ao viés do pesquisador, refere-se a qualquer tipo de influência negativa do conhecimento ou das suposições do pesquisador sobre o estudo. A formação, os valores e as pré-concepções dos pesquisadores influenciam as conclusões de qualquer estudo qualitativo. Uma avaliação criteriosa dos nossos resultados ajudou a mitigar essa ameaça.

Possíveis ameaças à confiabilidade, que se refere à consistência de certas medições; espera-se que as replicações do caso de estudo apresentem resultados semelhantes aos da Seção 4.2.1. É claro que os valores concretos medidos podem diferir dos apresentados aqui, pois são específicos dos participantes, mas as tendências e implicações subjacentes devem permanecer as mesmas.

6 CONSIDERAÇÕES FINAIS

A segurança de sistemas tem aumentado significativamente na última década, impulsionada pela necessidade de fornecer software seguro, reduzir o número de ataques bem-sucedidos e cumprir regulamentações e leis. No entanto, desenvolver software seguro não é uma tarefa fácil e frequentemente exige a experiência de especialistas em segurança. Esse desafio se intensifica quando são aplicadas práticas ágeis.

Este trabalho apresentou o SecurityRE, qual visa suportar o time ágil na derivação de requisitos de segurança. O método é composto por um metamodelo para servir como guia na especificação de estratégias de segurança em termos de detecção, recuperação e mitigação de ameaças. O intuito é melhorar a compreensão e documentação desses requisitos considerando a diferença de expertise e formação do time ágil. A validação piloto indicou percepções positivas quanto à clareza, utilidade e adequação do método ao desenvolvimento ágil. Por outro lado, dificuldades na finalização dos artefatos apontam para a melhorias na documentação do método.

Para integrar melhor o desenvolvimento de software seguro com técnicas e metodologias ágeis, a comunidade deve investir em esforços como o processo de derivação de requisitos de segurança. O objetivo é implementar o modelo, otimizá-lo ou sugerir abordagens semelhantes. Pretendemos estudar o desempenho do modelo em outros cenários do mundo real e coletar dados para refiná-lo. Dessa forma, buscamos aperfeiçoar o modelo para atingir seu objetivo: melhorar a comunicação entre a equipe de desenvolvimento, a equipe de segurança e a equipe ágil, permitindo que compartilhem ideias mais claras e colaborem de forma eficaz no desenvolvimento de software seguro.

REFERÊNCIAS

ABUKWAIK, H.; ZHANG, C. esquare: A formal methods enhanced square tool. In: . [S.l.: s.n.], 2012.

ALEXANDER, I. Misuse cases: use cases with hostile intent. *IEEE Software*, IEEE Computer Society, v. 20, p. 58–66, 1 2003. ISSN 0740-7459. Disponível em: ⟨http://ieeexplore.ieee.org/document/1159030/⟩.

ALTHONAYAN, A.; ANDRONACHE, A. Shifting from information security towards a cybersecurity paradigm. In: *ACM International Conference Proceeding Series*. [S.l.]: Association for Computing Machinery, 2018. p. 68–79. ISBN 9781450364898.

ARDO, A. A.; BASS, J. M.; GABER, T. An empirical investigation of agile information systems development for cybersecurity. In: *Lecture Notes in Business Information Processing*. [S.l.]: Springer Science and Business Media Deutschland GmbH, 2022. v. 437 LNBIP, p. 567–581. ISBN 9783030959463. ISSN 18651356.

ARDO, A. A.; BASS, J. M.; GABER, T. Towards secure agile software development process: A practice-based model. In: *Proceedings - 48th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2022*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2022. p. 149–156. ISBN 9781665461528.

ASTHANA, V. et al. *Practical Security Stories and Security Tasks for Agile Development Environments*. [S.l.], 2012. Disponível em: ⟨https://safecode.org/publication/SAFECode_Agile_Dev_Security0712.pdf⟩.

BAKER, O.; NGUYEN, Q. *A Novel Approach to Secure Microservice Architecture from OWASP vulnerabilities*. 2019. Acesso em: 04 setembro 2024.

BERNSMED, K. et al. Adopting threat modelling in agile software development projects. *Journal of Systems and Software*, Elsevier Inc., v. 183, 1 2022. ISSN 01641212.

BEZERRA, C. M. M.; SAMPAIO, S. C.; MARINHO, M. L. Secure agile software development: Policies and practices for agile teams. In: *Communications in Computer and Information Science*. [S.l.]: Springer Science and Business Media Deutschland GmbH, 2020. v. 1266 CCIS, p. 343–357. ISBN 9783030587925. ISSN 18650937.

BEZNOSOV, K.; KRUCHTEN, P. Towards agile security assurance. In: *Proceedings of the 2004 workshop on New security paradigms - NSPW '04*. ACM Press, 2005. p. 47. ISBN 1595930760. Disponível em: ⟨http://portal.acm.org/citation.cfm?doid=1065907.1066034⟩.

Common Criteria Recognition Arrangement. *Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model*. Revision 1. [S.l.], 2022. Acesso em: 01 janeiro 2025. Disponível em: ⟨https://commoncriteriaportal.org/files/ccfiles/CC2022PART1R1.pdf⟩.

GHARIB, M. Us4usec: A user story model for usable security. In: *Lecture Notes in Business Information Processing*. [S.l.]: Springer Science and Business Media Deutschland GmbH, 2024. v. 513, p. 257–272. ISBN 9783031594649. ISSN 18651356.

GLINZ, M.; WIERINGA, R. Guest editors' introduction: Stakeholders in requirements engineering. *IEEE Software*, v. 24, p. 18–20, 3 2007. ISSN 0740-7459. Disponível em: ⟨http://ieeexplore.ieee.org/document/4118646/⟩.

GOERTZEL, K. et al. *Software Security Assurance: A State-of-Art Report (SAR)*. 2007.

HANNOUSSE, A.; YAHIOUCHE, S. Securing microservices and microservice architectures: A systematic mapping study. *Computer Science Review*, Elsevier Ireland Ltd, v. 41, p. 100415, 8 2021. ISSN 15740137. Disponível em: ⟨https://linkinghub.elsevier.com/retrieve/pii/S1574013721000551⟩.

HUMAYUN, M. et al. Cyber security threats and vulnerabilities: A systematic mapping study. *Arabian Journal for Science and Engineering*, Springer, v. 45, p. 3171–3189, 4 2020. ISSN 21914281.

International Organization for Standardization. *ISO 13485:2016 — Medical devices — Quality management systems — Requirements for regulatory purposes*. [S.l.]: ISO, 2016. ⟨https://www.iso.org/standard/59752.html⟩. Acesso em: 25 de Março 2025.

International Organization for Standardization. *ISO/IEC 81001-5-1:2021 — Health software and health IT systems safety, effectiveness and security — Part 5-1: Security — Activities in the product life cycle*. [S.l.]: ISO, 2021. ⟨https://www.iso.org/standard/76097.html⟩. Acesso em: 25 de Março 2025.

JANISAR, A. A. et al. Security requirements assurance: An assurance case perspective. In: . [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2023. p. 78–83. ISBN 9798350310931.

JEDLITSCHKA, A.; CIOLKOWSKI, M.; PFAHL, D. Reporting Experiments in Software Engineering. In: Guide to advanced empirical software engineering. In: _____. [S.l.]: Springer, 2008. p. 201–228.

KORDY, B.; PIèTRE-CAMBACéDèS, L.; SCHWEITZER, P. *DAG-based attack and defense modeling: Don't miss the forest for the attack trees*. 2014. 1-38 p. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1574013714000100⟩.

KOTENKO, I. et al. Cyber attacker profiling for risk analysis based on machine learning. *Sensors*, MDPI, v. 23, 2 2023. ISSN 14248220.

LUNDGREN, B.; MöLLER, N. Defining information security. *Science and Engineering Ethics*, Springer Netherlands, v. 25, p. 419–441, 4 2019. ISSN 14715546.

MCDERMOTT, J.; FOX, C. Using abuse case models for security requirements analysis. In: *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)*. IEEE Comput. Soc, 1999. p. 55–64. ISBN 0-7695-0346-2. Disponível em: ⟨http://ieeexplore.ieee.org/document/816013/⟩.

MIHELIč, A.; VRHOVEC, S.; HOVELJA, T. Agile development of secure software for small and medium-sized enterprises. *Sustainability*, MDPI, v. 15, p. 801, 1 2023. ISSN 2071-1050. Disponível em: ⟨https://www.mdpi.com/2071-1050/15/1/801⟩.

MOHAMMAD, M. N. A.; NAZIR, M.; MUSTAFA, K. A systematic review and analytical evaluation of security requirements engineering approaches. *Arabian Journal for Science and Engineering*, v. 44, p. 8963–8987, 11 2019. ISSN 2193-567X. Disponível em: ⟨http://link.springer.com/10.1007/s13369-019-04067-3⟩.

MOYON, F. et al. Security compliance in agile software development: A systematic mapping study. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2020. p. 413–420. ISBN 978-1-7281-9532-2. Disponível em: ⟨https://ieeexplore.ieee.org/document/9226365/⟩.

NINA, H.; POW-SANG, J. A.; VILLAVICENCIO, M. Systematic mapping of the literature on secure software development. *IEEE Access*, Institute of Electrical and Electronics Engineers Inc., v. 9, p. 36852–36867, 3 2021. ISSN 21693536.

NäGELE, S.; SCHENK, N.; MATTHES, F. The current state of security governance and compliance in large-scale agile development: A systematic literature review and interview study. In: *Proceedings - 2023 IEEE 25th Conference on Business Informatics, CBI 2023*. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2023. ISBN 9798350315158.

NäGELE, S.; WATZELT, J.-P.; MATTHES, F. Investigating the current state of security in large-scale agile development. In: _____. [s.n.], 2022. p. 203–219. Disponível em: ⟨https://link.springer.com/10.1007/978-3-031-08169-9_13⟩.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering. *Information and software technology, 2015*, 2015.

PFLEEGER, S. L. A framework for security requirements. *Computers & Security*, v. 10, p. 515–523, 10 1991. ISSN 01674048. Disponível em: ⟨https://linkinghub.elsevier.com/retrieve/pii/016740489190076P⟩.

RASHEED, A. et al. *Requirement Engineering Challenges in Agile Software Development*. [S.l.]: Hindawi Limited, 2021.

REID, R.; NIEKERK, J. V. From information security to cyber security cultures. In: *2014 Information Security for South Africa*. IEEE, 2014. p. 1–7. ISBN 978-1-4799-3384-6. Disponível em: ⟨http://ieeexplore.ieee.org/document/6950492/⟩.

REZNIK, L. Malware and vulnerabilities detection and protection. In: _____. *Intelligent Security Systems: How Artificial Intelligence, Machine Learning and Data Science Work For and Against Computer Security*. [S.l.: s.n.], 2022. p. 177–246.

RINDELL, K. et al. Security in agile software development: A practitioner survey. *Information and Software Technology*, Elsevier B.V., v. 131, p. 106488, 3 2021. ISSN 09505849. Disponível em: ⟨https://linkinghub.elsevier.com/retrieve/pii/S0950584920302305⟩.

RODRIGUES, E. et al. *A Platform for Remote Monitoring of Older Adults: The Value of Heart Rate Variability*. 2022. Disponível em: ⟨https://www.researchsquare.com/article/rs-2059934/v1⟩.

RØSTAD, L. An extended misuse case notation: Including vulnerabilities and the insider threat. In: . [s.n.], 2006. Disponível em: ⟨https://api.semanticscholar.org/CorpusID:17678382⟩.

SALINI, P.; KANMANI, S. Survey and analysis on security requirements engineering. *Computers & Electrical Engineering*, v. 38, p. 1785–1797, 11 2012. ISSN 00457906. Disponível em: ⟨https://linkinghub.elsevier.com/retrieve/pii/S0045790612001644⟩.

SANTOS, O. *Cisco cyberops associate CBROPS 200-201 official cert guide*. Cisco Press, 2021. Acesso em: 15 de Março 2025. ISBN 9780136807834. Disponível em: ⟨https://elhacker.info/manuales/Hacking%20y%20Seguridad%20informatica/Cisco%20CyberOps/Cisco.Press.Cisco.CyberOps.Associate.CBROPS.200-201.Official.Cert.Guide..pdf⟩.

SCHIELE, N. D.; GADYATSKAYA, O. A novel approach for attack tree to attack graph transformation: Extended version. 10 2021. Disponível em: ⟨http://arxiv.org/abs/2110.02553⟩.

SOMMERVILLE, I. *Software engineering*. [S.l.]: Pearson, 2016. 796 p. ISBN 9780133943030.

SUBHA, R.; HALDORAI, A. An efficient identification of security threats in requirement engineering methodology. *Computational Intelligence and Neuroscience*, Hindawi Limited, v. 2022, 2022. ISSN 16875273.

TØNDEL, I. A.; JAATUN, M. G. Towards a conceptual framework for security requirements work in agile software development. In: *Research Anthology on Agile Software, Software Development, and Testing*. [S.l.]: IGI Global Scientific Publishing, 2022. p. 247–279.

United States. *Health Insurance Portability and Accountability Act of 1996 (HIPAA)*. 1996. Public Law 104–191. Disponível em: ⟨https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html⟩.

VALDéS-RODRíGUEZ, Y. et al. Towards the integration of security practices in agile software development: A systematic mapping review. *Applied Sciences*, v. 13, p. 4578, 4 2023. ISSN 2076-3417. Disponível em: ⟨https://www.mdpi.com/2076-3417/13/7/4578⟩.

VENTER, H. S. et al. *ISSA : 2014 Information Security for South Africa : proceedings of the ISSA 2014 conference : 13-14 August 2014, Radisson Blu Gautrain Hotel, Sandton, Johannesburg, South Africa*. [S.l.]: IEEE, 2014. ISBN 9781479933839.

WEISS, J. D. A system security engineering process. In: *Proceedings of the 14th National Computer Security Conference*. [s.n.], 1991. v. 249, p. 572–581. Acesso em: 20 abril 2024. Disponível em: ⟨https://csrc.nist.gov/pubs/conference/1991/10/01/proceedings-14th-national-computer-security-confer/final⟩.

WIERINGA, R. *Design Science Methodology for Information Systems and Software Engineering*. [S.l.]: Springer Berlin Heidelberg, 2014.

WIERINGA, R. J. *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, 2014. v. 24. 45-77 p. ISSN 07421222. ISBN 978-3-662-43838-1. Disponível em: ⟨https://link.springer.com/10.1007/978-3-662-43839-8⟩.

WOHLIN, C. et al. *Experimentation in Software Engineering*. [S.l.]: Springer, 2012.

APÊNDICE A - Uma Pesquisa sobre Abordagens para Especificar Requisitos de Segurança

Uma Pesquisa sobre Abordagens para Especificar Requisitos de Segurança

Edilson Costa
edilson.costa@aluno.uepb.edu.br
Universidade Estadual da Paríba, Brasil

Joyce Lima
joyce.avelino@aluno.uepb.edu.br
Universidade Estadual da Paríba, Brasil

Vinícius Alves
jose.vinicius.alves@aluno.uepb.edu.br
Universidade Estadual da Paríba, Brasil

RESUMO

O trabalho trata sobre os métodos utilizados na especificação de requisitos de segurança em engenharia de software, em específico, sobre quais são os métodos que possuem ferramentas. A segurança é essencial para garantir a integridade, confidencialidade e disponibilidade dos sistemas. O estudo identifica e analisa oito métodos principais: Misuse Cases, Abuse Cases, Secure Tropos, Extended Misuse Case Notation, SQUARE, STORE, US4USec e SecureUML, destacando suas principais características e as ferramentas que oferecem suporte à aplicação prática dessas metodologias. A análise revelou que, embora muitos métodos se baseiam em notações conhecidas como UML, poucos dispõem de ferramentas automatizadas dedicadas. A escolha do método mais adequado depende do contexto do projeto, da criticidade do sistema e do nível de conhecimento da equipe em segurança. Por fim, este Survey pode contribuir para futuras pesquisas sobre engenharia de requisitos de segurança ao mapear e sintetizar as principais metodologias.

PALAVRAS-CHAVE

Segurança, Requisitos de Segurança, Especificação de Requisitos de Segurança.

ACM Reference Format:
Edilson Costa, Joyce Lima, and Vinícius Alves. 2024. Uma Pesquisa sobre Abordagens para Especificar Requisitos de Segurança. In *38th Brazilian Symposium on Software Engineering (SBES '24), Junho, 04, 2024, Campina Grande, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.11111/111111111.11111111

1 INTRODUÇÃO

É inegável que ao longo dos anos, a indústria de software tem crescido cada vez mais, impactando diretamente no dia a dia da sociedade. Os softwares são uma importante ferramenta para as mais diversas áreas, sendo praticamente impossível encontrar quem não faz seu uso. O mundo foi modernizado, sendo imerso pelo vasto uso de aplicações digitais, desde a comunicação e saúde até a educação e entretenimento, facilitando processos e otimizando rotinas.

Ainda assim, a má operação e construção de um software pode levar a problemas como perdas financeiras, roubo de dados, invasão de privacidade e até mesmo risco à vida humana. Dessa forma, a rápida expansão, proporção e conectividade que os sistemas podem chegar, bem como a grande diversidade de tecnologias que podem ser utilizadas em seu desenvolvimento, como redes de computadores, IoT, computação na nuvem e Inteligência artificial, fez com que a segurança de software se tornasse uma nova preocupação.

E para assegurar a segurança de aspectos tanto humanos quanto tecnológicos, os Métodos de Segurança de Engenharia de Software (SSE) podem ser utilizados. Eles estão relacionados à criação de Softwares resistentes a ataques, bem como capazes de manter sua integridade e privacidade [9]. E dentro dessa área, os requisitos de segurança tem um papel fundamental em proporcionar métodos e técnicas para lidar com essa tarefa desde o primórdio do ciclo de vida do software [16].

Contudo, a segurança ainda é vista por muitas empresas como uma fase que deve ser feita após o processo de desenvolvimento do Software, sendo muitas vezes negligenciadas no pré-processo de criação [17]. E essa falta de conscientização sobre o desenvolvimento seguro de software representa um dos maiores riscos enfrentados pelas organizações [11]. Esse tipo de negligência pode resultar em prejuízos financeiros significativos além do risco de processos judiciais pelos clientes prejudicados pela perda de dados, decorrente de vulnerabilidades introduzidas pela ausência de práticas de segurança durante o desenvolvimento.

Diante do exposto, há a necessidade de ter como foco, a segurança de um software desde o início do seu desenvolvimento. Dessa forma, este estudo tem como objetivo realizar uma análise das distintas abordagens propostas para a derivação de requisitos de segurança, com o intuito de subsidiar futuras pesquisas e práticas no campo da engenharia de requisitos de segurança. Para atingir o objetivo mencionado acima, a seguinte pergunta de pesquisa foi formulada:

QP01. Quais os métodos de derivação de requisitos de segurança que são suportados por ferramentas?

Além disso, o trabalho tem a seguinte estrutura: na seção 2 são vistos os conceitos necessários para o entendimento e desenvolvimento da pesquisa, e, na seção 3, os trabalhos relacionados que embasaram a pesquisa. Na seção 4, está a metodologia e critérios de avaliação das abordagens escolhidas. Na seção 5, é visto os métodos para especificar requisitos de segurança. Já a seção 6 conta com os resultados obtidos, em que a pergunta de pesquisa é respondida. E, por fim, na seção 7 ficam as discussões e na seção 8 as conclusões do trabalho.

UEPB '1, Junho, 04, 2025, Campina Grande, Brazil

ACM ISBN 111-1-1111-1111-1/10/01…$15.00
https://doi.org/10.11111/111111111.11111111

2 SEGURANÇA DE SOFTWARE

Nesta seção serão listados alguns conceitos sobre segurança de software para embasar as próximas seções.

2.1 Conceitos de segurança de software

O conceito do que é segurança de software tem sido descrito por diversos autores ao longo dos anos. Sendo algumas dessas definições as seguintes:

- "Software security refers to the processes and practices involved in developing secure software systems that are resistant to malicious attacks and unintended vulnerabilities [18], [6]."
- "Software security is about building secure software which includes designing software to be secure, making sure that the software is secure, and educating software developers and users about how to build secure things [14]."
- "Defends against software exploits by building software to be secure in the first place, mostly by getting the design right (which is hard) and avoiding common mistakes [5]."

Apesar de muitos autores proporem definições para a segurança de software, não houve consenso sobre uma definição padrão. Nota-se que a maioria das definições destacadas enfatiza a criação de software seguro, em vez de simplesmente o proteger. Construir um software seguro significa projetá-lo e implementá-lo com foco na segurança desde o início do desenvolvimento, enquanto proteger o software está relacionado a criar o sistema primeiro para depois aplicar mecanismos de segurança na tentativa de o tornar seguro [9].

2.2 Requisitos de segurança de software

De acordo com diversos estudos, aproximadamente 80% dos erros e falhas em softwares têm origem em requisitos inadequados coletados durante o processo de levantamento de requisitos [21]. Em documentos de especificação de requisitos, é frequente a apresentação dos requisitos de segurança em seções específicas, frequentemente replicados a partir de conjuntos genéricos previamente estabelecidos. Esses requisitos, em sua maioria, envolvem a utilização de mecanismos tradicionais, como autenticação por senha, autorização de usuários, implementação de firewalls, controle de acesso e ferramentas de detecção de vírus. Sendo, de modo geral, observado que os requisitos funcionais do sistema, sob a ótica do usuário, recebem maior ênfase durante o processo de desenvolvimento, enquanto os requisitos de segurança tendem a ser abordados de forma secundária.

Ademais, foi notado que alguns engenheiros de requisitos possuem um conhecimento limitado em extração, análise e especificação de requisitos de segurança, como resultado, especificam restrições de arquitetura e design, em vez de definir os reais requisitos de segurança. E, tendo em vista que os requisitos são originados da linguagem natural, a Engenharia de Requisitos (RE) enfrenta desafios tanto na elicitação quanto na documentação de requisitos de segurança, uma vez que há uma tendência à interpretação inadequada das necessidades reais e dos conceitos de segurança empregados[7].

A negligência na consideração de requisitos de segurança durante a fase de levantamento de requisitos, ou a tentativa de incorporá-los apenas após a conclusão do projeto do sistema, frequentemente resulta em vulnerabilidades significativas. Requisitos de segurança mal elaborados impactam negativamente fatores como o custo, o tempo de desenvolvimento, a necessidade de retrabalho e a frequência de alterações nos requisitos [19]. E, por conta de requisitos de segurança deficientes, vulnerabilidades em sistemas de software podem colocar indivíduos e empresas em risco de várias maneiras.

Diante da problemática evidenciada, torna-se essencial a adoção de metodologias específicas para a derivação de requisitos de segurança e da integração desses requisitos desde as fases iniciais do desenvolvimento. Nesse contexto, o presente trabalho tem como objetivo investigar e analisar as abordagens existentes para a especificação de requisitos de segurança, em especial, se elas possuem ferramentas de auxílio, de modo a contribuir para a melhoria das práticas adotadas na engenharia de software.

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados os principais artigos que serviram de base para o desenvolvimento da pesquisa.

O "Systematic Mapping Study on Security Approaches in Secure Software Engineering" teve como principal objetivo, estudar medidas de segurança no contexto do desenvolvimento de software seguro durante o estudo de mapeamento sistemático [9]. Nele, 116 estudos foram analisados e classificados com base nos métodos utilizados, fases do SDLC cobertas, e evidências empíricas. Além disso, foram destacados métodos como o Secure Tropos, Misuse Case, SQUARE, e SecureUML, onde foram abordadas a força, fraqueza, oportunidades e ameaças de cada método. Contudo, embora o estudo tenha fornecido uma visão ampla das abordagens existentes, o foco está mais na categorização e frequência das soluções do que em sua aplicabilidade prática com ferramentas de apoio.

Adicionalmente, o "Systematic Mapping of the Literature on Secure Software Development", que também se caracteriza como um mapeamento sistemático, teve como foco fornecer uma visão geral das tendências de desenvolvimento de software seguro para ajudar a identificar tópicos que já foram extensivamente estudados e aqueles que ainda precisam ser considerados [20]. Para isso, o estudo examinou e organizou 528 artigos de acordo com cada fase do ciclo de vida de desenvolvimento de software e tipos de métodos. Como resultado, foram vistos os métodos mais populares e usados, como o misuse case, assim como as contribuições mais populares reportadas na literatura.

O "Systematic Literature Review on Security Risks and Its Practices in Secure Software Development" é uma revisão sistemática da literatura que tem como objetivo "aprender sobre riscos e práticas de segurança de software para que métodos seguros de desenvolvimento de software possam ser melhor projetados" [8]. Nessa pesquisa, um total de 121 trabalhos foram selecionados, em que foram identificados 142 riscos e 424 boas práticas na engenharia de requisitos de segurança, em cada fase do ciclo de vida de desenvolvimento de um software. Dentre essas práticas, os mesmos citam métodos como SecureUML, Abuse Cases, Secure Tropos, entre outros que podem ser utilizados na etapa de engenharia dos requisitos de segurança. Com os resultados alcançados, os autores buscam

auxiliar as empresas e desenvolvedores a aprimorarem os níveis de segurança dos softwares desenvolvidos, aumentando a conscientização dos desenvolvedores sobre práticas de desenvolvimento seguras.

Por fim, apesar de fornecerem uma visão abrangente da área, os estudos priorizam a identificação dos métodos para especificar requisitos de segurança e sua distribuição na literatura, sem explorar com profundidade o nível de suporte ferramental disponível para cada uma, sendo essa, a principal diferença entre o presente trabalho e as pesquisas já existentes que foram utilizadas como base para o survey. Nesse contexto, o trabalho faz uma análise entre oito métodos voltados à especificação de requisitos de segurança, examinando não apenas seus fundamentos teóricos, mas também investigando se tais métodos contam com ferramentas que apoiem sua implementação prática.

4 METODOLOGIA

Este estudo caracteriza-se como um survey bibliográfico, modalidade de pesquisa que visa reunir, organizar, interpretar e analisar criticamente publicações científicas sobre determinado tema. De acordo com Gil (2008)[12], a pesquisa bibliográfica se baseia na análise de materiais já publicados, como artigos científicos, livros, anais de eventos e teses, permitindo a identificação de tendências, lacunas e avanços em uma área de conhecimento. Trata-se, portanto, de um método essencial para mapear o estado da arte de um campo específico da ciência.

Segundo Groves et al. (2009)[11], surveys podem ser utilizados como estratégia metodológica para fundamentar investigações exploratórias ou descritivas, quando se deseja compreender, categorizar e comparar fenômenos a partir de fontes já existentes. No presente estudo, o survey bibliográfico teve como foco identificar os métodos utilizados para a especificação de requisitos de segurança em software e, especialmente, verificar quais desses métodos possuem suporte por ferramentas, respondendo à seguinte pergunta de pesquisa:

- QP01: Quais os métodos para especificação de requisitos de segurança que possuem suporte por ferramentas?

4.1 Estratégia de busca

A coleta dos dados foi realizada por meio de buscas manuais nas seguintes bases acadêmicas:

- IEEE Xplore
- Scopus
- Google Scholar

As buscas foram realizadas com o auxílio de operadores booleanos e uso de palavras-chave, como: ((“software security” OR “secure software requirements” OR “security requirements methods”) AND (“Software Engineering” OR “Software Development lifecycle”))

O objetivo dessa busca foi maximizar o número de trabalhos que abordassem, direta ou indiretamente, especificação de requisitos de segurança, seus métodos e frameworks voltados à segurança no ciclo de vida de desenvolvimento de software.

4.2 Critério de seleção dos trabalhos

A partir das strings de busca, a seleção dos trabalhos foi orientada com o objetivo de assegurar a pertinência temática e a adequação dos estudos ao escopo da investigação. Os critérios a seguir visam filtrar as publicações mais relevantes para a análise dos métodos de especificação de requisitos de segurança em software, garantindo consistência à pesquisa.

4.2.1 Critérios de inclusão.

- CI1: Artigos na área de segurança de software;
- CI2: Artigos relacionados ao SDLC;
- CI3: Textos disponíveis em inglês ou português, com acesso integral ao conteúdo.

4.2.2 Critérios de exclusão.

- CE1: Trabalhos não relacionados diretamente ao tema central;
- CE2: Trabalhos que não estavam disponíveis em inglês ou português;
- CE3: Trabalhos indisponíveis para acesso completo ou com conteúdo insuficiente para análise metodológica.

5 MÉTODOS PARA ESPECIFICAR REQUISITOS DE SEGURANÇA

5.1 Misuse Case

Os Misuse Cases [2] consistem em uma técnica especializada que se concentra na identificação e análise de potenciais ameaças ou cenários indesejados que podem vir a impactar ou comprometer o sistema. Diferente dos tradicionais casos de uso que descrevem funcionalidades esperadas, os “misuse cases” assumem a perspectiva de agentes “maliciosos” e ajudam os engenheiros a prever ataques e vulnerabilidades. Essa abordagem contribui diretamente para a criação de sistemas mais seguros e resilientes.

Essa técnica envolve etapas como a identificação de agentes maliciosos, a elaboração de cenários de ataque e a análise de como esses cenários ameaçam as funcionalidades legítimas do sistema. Os misuse cases podem se relacionar com casos de uso tradicionais de duas formas: Ameaça, Indicando como um caso de uso indevido pode comprometer um determinado caso de uso regular, e Mitigação, quando medidas ou controles de segurança podem ser implementados para neutralizar os efeitos de um caso de uso indevido.

O processo de modelagem de “misuse cases” começa pela identificação de atores, legítimos e maliciosos, e dos casos de uso normais. Em seguida, analisam-se possíveis ações maliciosas que exploram vulnerabilidades, detalhando o objetivo do ataque e como ele seria executado.

5.2 Abuse Case

Abuse Case [13] é uma técnica utilizada na engenharia de software e na segurança da informação para determinar interações entre um sistema e uma ou mais entidades que causam dano, seja ao próprio sistema, a um ator legítimo ou a uma parte interessada. Diferente dos casos de uso, que descrevem o comportamento desejado do sistema, e dos misuse cases, que focam na intenção maliciosa e no uso indevido potencial, os abuse cases destacam a interação completa que de fato resulta em um dano.

Em termos de estrutura, os abuse cases seguem um formato semelhante ao dos casos de uso tradicionais e podem ser representados por diagramas UML e descrições em linguagem natural. No entanto, seu foco está na modelagem de comportamentos maliciosos e suas consequências. Cada abuse case geralmente envolve:

- O ator malicioso, com informações detalhadas sobre seus objetivos, recursos ferramentas ou colaboradores, e habilidades técnicas;
- A interação nociva completa, ou seja, não apenas uma tentativa de ataque, mas uma que resulte em dano real, por exemplo, o simples vazamento de uma chave de sessão não constitui um abuse case, mas o uso dessa chave para manipular ou expor dados sensíveis, sim;
- O tipo e a faixa de privilégios abusados, identificando tanto o nível mínimo de privilégio necessário para realizar o ataque quanto o nível máximo considerado no modelo;
- O dano resultante, descrito em termos familiares ao domínio de negócio ou ao usuário, para comunicar claramente o impacto às partes interessadas.

Os abuse cases podem ser organizados em estruturas de árvore ou em DAGs (Grafos Acíclicos Dirigidos), semelhantes às árvores de ataque. Nessa estrutura, a raiz representa o sistema, os ramos representam os recursos ou componentes alvo, e os caminhos representam os subsistemas ou funcionalidades exploradas para realizar o ataque. Isso ajuda a ilustrar vetores alternativos de ataque e diferentes níveis de escalonamento de privilégios ou comprometimento do sistema. O processo de modelagem de abuse cases geralmente envolve quatro etapas principais:

- Identificação de atores maliciosos: Com base no modelo de casos de uso, definem-se versões maliciosas de quaisquer papéis que possam agir de forma prejudicial.
- Devem ser incluídas tanto ameaças internas, quanto ameaças externas, classificadas por seus recursos e nível de expertise.
- Identificação dos casos de abuso: Para cada ator malicioso, definem-se as interações prejudiciais com o sistema. Nomeia-se cada abuse case e, se necessário, criam-se os diagramas correspondentes.
- Verificação da granularidade: Garante-se que o modelo mantenha níveis consistentes de detalhamento. Buscando evitar incluir muitos casos semelhantes que diferem apenas em aspectos triviais.
- Verificação de completude e minimalidade: Cada abuse case deve descrever claramente o dano, e o modelo não deve negligenciar ameaças críticas.

Essa técnica contribui para a antecipação de riscos concretos e a definição de requisitos de segurança mais precisos e eficazes.

5.3 Secure Tropos Methodology

O Secure Tropos é uma extensão da metodologia de Engenharia de Software Baseada em Agentes, adaptada ao sistema futuro e seu ambiente organizacional, chamada Tropos. Essa extensão serve para modelar e analisar requisitos de segurança em conjunto com requisitos funcionais e fornece um processo de análise de requisitos para orientar os desenvolvedores de sistemas desde a aquisição até a validação. [12]

A metodologia estrutura o processo de especificação em três etapas fundamentais para representar um modelo de requisitos [12]:

- Modelagem de Atores: Envolve a identificação e análise das partes interessadas no domínio e dos atores do sistema, considerando seus objetivos, direitos e habilidades.
- Modelagem de Confiança: Refere-se à representação das expectativas que os atores têm em relação ao desempenho e ao comportamento adequado de outros atores, seja em relação a um objetivo, tarefa ou recurso. Essas expectativas são expressas por meio de confiança na execução e confiança nos links de permissão.
- Modelagem de Dependência de Execução: Consiste em identificar a transferência de responsabilidades entre atores, representada por meio de delegação de links de execução.
- Modelagem de Delegação de Permissão: Envolve a transferência de autoridade entre atores, modelada por meio de delegação de links de permissão.

5.4 An extended misuse case notation

A notação de Extended Misuse Case (Casos de Uso Indevido Estendidos) é uma evolução da notação tradicional, desenvolvida para visualizar explicitamente ameaças internas e explorar as fragilidades do projeto de engenharia no que diz respeito aos requisitos de segurança. A versão original da notação focava exclusivamente em agressores externos e nas ameaças que eles representam aos usos legítimos do sistema. No entanto, essa abordagem era limitada, pois ignorava um vetor crítico de ameaça: usuários internos autorizados que podem agir de forma maliciosa.

Com essa nova notação, insiders (usuários internos maliciosos, como funcionários, desenvolvedores ou usuários com privilégios) e vulnerabilidades (falhas que os atacantes podem explorar) agora podem ser representados graficamente. Além disso, uma nova relação chamada exploração foi introduzida, conectando a ameaça à vulnerabilidade que ela explora, facilitando a visualização do caminho desde a ameaça até sua concretização.

Definições-chave:

- Insider (Ameaça Interna): Um usuário malicioso que faz parte de um grupo com acesso a um sistema ou organização, como um funcionário ou membro da equipe de desenvolvimento.
- Vulnerabilidade: Uma fraqueza no sistema que pode ser explorada por um usuário malicioso para realizar um ataque.

5.5 SQUARE

SQUARE [15] (Security Quality Requirements Engineering) é uma metodologia que foi desenvolvida por Nancy Meade, Donald Friesmith e Carol Woody. Seu principal objetivo é auxiliar as equipes de desenvolvimento na identificação, categorização e priorização de requisitos de segurança para sistemas e aplicações de TI.

Diferentemente das abordagens tradicionais, que frequentemente tratam os requisitos de segurança como uma consideração secundária, a abordagem SQUARE integra a segurança desde as etapas iniciais do ciclo de vida do desenvolvimento. Isso permite uma abordagem mais eficiente para incorporar proteção contra ameaças e vulnerabilidades de forma estruturada e alinhada aos objetivos do negócio.

A metodologia também é útil para sistemas já existentes, pois permite a documentação e análise de aspectos de segurança e fornece orientações para melhorias e adaptações futuras. O SQUARE consiste em nove etapas sequenciais e estruturadas, desde a definição de conceitos comuns até a verificação final dos requisitos priorizados. Essas ações envolvem a participação ativa das partes interessadas (clientes, usuários, representantes de organizações) e de solicitantes com conhecimento na área de segurança. As atividades realizadas incluem: Estabelecimento de definições acordadas;

- Identificação dos objetivos de segurança;
- Desenvolvimento de artefatos de apoio (como cenários e casos de uso indevido);
- Avaliação de riscos;
- Seleção de técnicas de elicitação;
- Coleta, categorização, priorização e inspeção dos requisitos.

O resultado final é um documento de requisitos de segurança bem fundamentado e verificável, acordado por todas as partes interessadas, servindo como base para o desenvolvimento de um sistema seguro e resiliente.

5.6 STORE

O método STORE [3] é uma abordagem sistemática composta por 10 etapas que orientam a engenharia de segurança desde os estágios iniciais do desenvolvimento de software. Sua estrutura foi concebida para integrar de forma eficaz os interesses de segurança de todas as partes interessadas relevantes, considerando ativamente os agentes de ameaça e as vulnerabilidades do sistema. Diferenciando-se por sua abordagem explicitamente orientada a ameaças, o STORE permite a geração de requisitos de segurança com base em riscos específicos previamente identificados.

A metodologia inicia-se com um brainstorming para a identificação dos objetivos do sistema, seguido da identificação e priorização das partes interessadas. Em seguida, os objetivos são consolidados com os stakeholders, garantindo alinhamento entre as expectativas e as necessidades de segurança. Posteriormente, procede-se à identificação dos ativos do sistema, que serão analisados na etapa de análise de ataques de segurança. Com base nessa análise, ocorre a identificação e classificação das ameaças, permitindo uma avaliação e priorização dos riscos.

A partir da compreensão dos riscos mais relevantes, realiza-se a elicitação dos requisitos de segurança, que são então validados para garantir sua consistência, relevância e viabilidade. Por fim, os requisitos validados são documentados formalmente, resultando em um artefato que servirá de base para o desenvolvimento seguro do sistema. Cada uma dessas etapas é interdependente e sequencial, tornando o processo linear e iterativo, com foco na integração da segurança desde as fases iniciais do ciclo de vida do software.

5.7 US4USec

O modelo US4USec [4] estende a estrutura tradicional de histórias de usuário "Como <tipo de usuário>, eu quero , para que <benefício>" com o intuito de integrar requisitos de segurança e aspectos de usabilidade da segurança. Ele é composto por duas partes principais: uma voltada para a especificação das funções e outra para os critérios de aceitação (AC).

Na seção de funções, o modelo permite a definição de até uma função principal (opcional), uma função de segurança obrigatória e uma ou mais funções USEC (usability of security). Essas funções são conectadas entre si para refletir a relação entre o que o sistema faz, como ele protege o usuário e como essa proteção é apresentada de forma clara e acessível.

Os critérios de aceitação são divididos em dois grupos: FSAC (Critérios de Aceitação Funcionais e de Segurança) e USECAC, que são baseados em heurísticas de usabilidade da segurança, como visibilidade, clareza e prevenção de erros. Cada diretriz é acompanhada de critérios específicos (USHC) que facilitam a avaliação prática.

O modelo diferencia claramente os problemas a serem resolvidos (funcionalidades) das condições para sua resolução (critérios de aceitação). A história é escrita prioritariamente pelo usuário, enquanto os critérios de segurança e usabilidade podem ser sugeridos por especialistas e posteriormente revisados pelo próprio usuário.

Desenvolvido de forma iterativa e fundamentado em práticas da literatura, o US4USec oferece uma estrutura clara e flexível para integrar segurança útil desde a fase de definição de requisitos, mesmo para usuários com pouca experiência técnica.

5.8 SecureUML

SecureUML [10] é uma linguagem de modelagem baseada em UML, projetada para integrar políticas de controle de acesso em um processo de desenvolvimento orientado por modelos. Baseado no modelo de controle de acesso baseado em papéis (RBAC – Role-Based Access Control), o SecureUML permite especificar permissões, papéis, atribuições de papéis e restrições de permissões diretamente em modelos UML.

Sua estrutura é suportada por um metamodelo extensível que define elementos como usuário, papel, permissão, restrição e conjunto de recursos. Em vez de tratar os recursos protegidos como tipos específicos, o modelo permite que qualquer elemento UML atue como fonte protegida. As permissões são associadas aos elementos do modelo por meio de ActionTypes, que representam ações como ler, escrever ou executar, utilizando uma semântica próxima ao vocabulário do domínio.

O modelo também oferece suporte a restrições de acesso expressas em OCL (Object Constraints Language) e pode gerar automaticamente infraestruturas de controle de acesso para plataformas como Enterprise Java Beans (EJB).

O SecureUML é implementado como um perfil UML, garantindo compatibilidade com ferramentas CASE e utilizando notação visual em UML, o que torna a modelagem mais acessível para profissionais que não são especialistas em segurança. Além disso, ele pode ser refinado para diferentes plataformas usando dialetos específicos, o que o torna adequado a diversos ambientes e permite a geração automática de componentes de segurança a partir do próprio modelo.

Como resultado, o SecureUML aumenta a produtividade no desenvolvimento de sistemas seguros e melhora a qualidade na implementação de políticas de controle de acesso, ao mesmo tempo que reduz inconsistências e erros comuns associados à configuração manual da segurança.

Método	Técnica Principal	Pontos em Comum	Ferramenta
SecureUML	UML + RBAC + OCL	Uso de UML, foco em controle de acesso, integra segurança à modelagem	Não
Abuse Cases	Casos de uso adaptados para ataques	Usa notação de casos de uso; fácil de entender; foco na análise de ameaças	Não
Misuse Cases	Extensão de casos de uso UML	Uso de atores hostis; relações "threatens" e "mitigates"; comunicação com stakeholders	Não
Extended Misuse Cases	Misuse Cases + Insiders + Vulnerabilidades	Estende misuse cases com atores internos e vulnerabilidades; foca em riscos reais e modelagem rica	Não
SQUARE	Processo sistemático de 9 etapas	Elicitação e priorização sistemática de requisitos; integração no início do ciclo de vida	Sim
STORE	Análise de ameaças e pontos de ataque	Foco na elicitação sistemática; orientação por ameaças; define PoA, PoB, PoC, PoD	Não
US4USec	User Stories adaptadas para segurança	Foco em requisitos de segurança usáveis; integração com histórias de usuário (NFR + user-centered)	Não
Secure Tropos	Extensão orientada à segurança do Tropos	Integra modelagem de requisitos sociais e técnicos; foco em análise formal e socio-organizacional	Sim

Figure 1: Tabela comparativa entre os métodos.

6 RESULTADOS

A análise comparativa realizada entre os oito métodos voltados à especificação de requisitos de segurança em sistemas de software evidencia uma certa diversidade de abordagens, técnicas e níveis de consolidação. A Tabela 1 sintetiza as características principais de cada um desses métodos, destacando seus fundamentos, pontos em comum e a existência ou não de ferramentas que apoiem sua aplicação prática.

De maneira geral, observa-se que a maioria dos métodos se apoia em notações conhecidas, como a UML (Unified Modeling Language), o que facilita sua compreensão e adoção por parte de desenvolvedores e analistas. Métodos como Abuse Cases, Misuse Cases e Extended Misuse Cases exemplificam essa tendência, utilizando ou adaptando casos de uso UML para expressar comportamentos indesejados, ameaças ou vulnerabilidades. Essa familiaridade contribui para a comunicação efetiva entre especialistas e não-especialistas em segurança.

Embora esses métodos não contem com ferramentas dedicadas ou automatizadas, todos compartilham a capacidade de poderem ser modelados em qualquer ferramenta UML tradicional, como Visual Paradigm, StarUML, Draw.io ou FlowChart. Isso é possível devido à sua aderência à notação de casos de uso e à simplicidade gráfica que os caracteriza. A mesma lógica se aplica ao SecureUML, que, apesar de possuir protótipos específicos para geração de infraestrutura de controle de acesso, também pode ser modelado em qualquer ambiente UML que aceite extensões.

Apesar dessa similaridade quanto ao suporte por ferramentas UML genéricas, existem diferenças conceituais relevantes entre os três métodos. Abuse Cases propõem a representação de cenários de ameaça como casos de uso abusivos, comumente modelados separadamente dos casos de uso funcionais. Misuse Cases integram ameaças e funcionalidades legítimas em um mesmo diagrama, permitindo a visualização de interações entre casos de uso e atores maliciosos. Já o modelo Extended Misuse Case avança ao incorporar a representação explícita de vulnerabilidades e a diferenciação entre atacantes externos e internos, aumentando a expressividade do modelo, mas também a complexidade de sua aplicação prática.

No que se refere ao suporte por ferramentas, identificou-se que o método SQUARE possui a ferramenta eSQUARE[1] , que fornece

suporte a todas as etapas da metodologia, auxiliando na sistematização e rastreabilidade das atividades realizadas. Partindo para a metodologia Secure Tropos, a mesma se destaca por contar com a ST-Tool, uma ferramenta robusta que oferece um ambiente gráfico para modelagem e integra recursos de verificação automática. Esse suporte de ferramentas torna o método mais aplicável em contextos que exigem maior rigor na verificação e validação dos requisitos de segurança.

7 DISCUSSÃO

Embora métodos com ferramentas dedicadas proporcionem maior automatização e suporte à análise formal, modelos mais acessíveis como misuse cases, abuse cases e STORE oferecem maior simplicidade e menor curva de aprendizado, ainda que resultem em uma menor escalabilidade.

Dessa forma, a escolha do método mais adequado depende do contexto do projeto, do grau de conhecimento da equipe ou do profissional responsável e dos objetivos de segurança visados. Modelos com ferramentas de suporte tendem a ser mais adequados em ambientes corporativos com requisitos de conformidade e alta criticidade, enquanto abordagens mais simples podem ser relevantes em fases iniciais ou em equipes com menor conhecimento em segurança.

Um dos desafios enfrentados neste trabalho foi a definição clara dos métodos a serem incluídos na análise. A área de engenharia de requisitos de segurança é marcada por uma sobreposição de abordagens, muitas das quais evoluem a partir de outras ou compartilham fundamentos semelhantes. Devido à limitação da metodologia adotada, não foi possível obter uma visão ampla da quantidade total de métodos existentes que contam ou não com ferramentas de suporte. Isso fez com que a análise ficasse restrita ao escopo selecionado, o que representa uma limitação importante a ser considerada quanto à generalização dos resultados.

Além disso, podemos observar que diversos métodos fazem uso da UML como base para modelagem; no entanto, muitos deles não contam com ferramentas específicas que auxiliem diretamente em seu desenvolvimento. Métodos como o STORE ainda são carentes de soluções que ofereçam suporte automatizado para todas as suas etapas. A ausência de ferramentas dedicadas pode impactar tanto na eficiência quanto na consistência da aplicação desses métodos. Portanto, o desenvolvimento de ferramentas de apoio não só facilitaria a adoção e aplicação desses métodos, como também poderia contribuir significativamente para a qualidade e repetibilidade do processo de engenharia de requisitos de segurança.

8 CONCLUSÃO

De maneira geral, nesse estudo, observou-se que grande parte dos métodos abordados utiliza notações estabelecidas, como a UML. Constatou-se que poucos desses métodos contam com ferramentas automatizadas dedicadas. Na maioria dos casos, os profissionais dependem de ferramentas genéricas de modelagem para realizar a aplicação prática dessas metodologias.

Além disso, a análise também mostrou que a escolha do método deve considerar o quão crítico é o sistema, o conhecimento da equipe em engenharia de segurança e os recursos disponíveis. Modelos mais simples podem ser aplicados com maior facilidade em contextos menos críticos ou em fases iniciais do desenvolvimento, enquanto metodologias que contam com suporte automatizado e formalização mais robusta são mais indicadas para ambientes regulados ou que exigem alta garantia de segurança.

Esse estudo colabora para o avanço da engenharia de requisitos de segurança ao mapear e sintetizar as principais metodologias disponíveis, bem como ao oferecer uma visão sobre o suporte por ferramentas que cada uma oferece. A expectativa é que os resultados sirvam como base para pesquisas futuras que podem se aprofundar nas ferramentas que cada método utiliza, e para aquelas que não possuem, compreender de forma aprofundada seu funcionamento prático.

REFERENCES

[1] Hadil Abukwaik and Cui Zhang. 2012. eSQUARE: A Formal Methods Enhanced SQUARE Tool. *Proceedings of the 2012 International Conference on Software Engineering Research and Practice.*

[2] Ian Alexander. 2003. Misuse cases: Use cases with hostile intent. *IEEE Software* 20 (2003), 58–66. Issue 1. https://doi.org/10.1109/MS.2003.1159030

[3] Md Tarique Jamal Ansari, Dhirendra Pandey, and Mamdouh Alenezi. 2022. STORE: Security Threat Oriented Requirements Engineering Methodology. *Journal of King Saud University - Computer and Information Sciences* 34 (2 2022), 191–203. Issue 2. https://doi.org/10.1016/j.jksuci.2018.12.005

[4] Mohamad Gharib. 2024. US4USec: A User Story Model for Usable Security. In *Lecture Notes in Business Information Processing*, Vol. 513. Springer Science and Business Media Deutschland GmbH, 257–272. https://doi.org/10.1007/978-3-031-59465-6_16

[5] G. Hoglund and Gary McGraw. 2004. Exploiting Software: How to Break Code. (01 2004).

[6] IEEE Computer Society. 2025. Software Security. https://www.computer.org/resources/software-security. Acessado em: 12 de maio de 2025.

[7] Massila Kamalrudin, Nuridawati Mustafa, and Safiah Sidek. 2018. *A Template for Writing Security Requirements.* 73–86. https://doi.org/10.1007/978-981-10-7796-8_6

[8] Dr-Rafiq Khan, Siffat Ullah Khan, Habib Khan, and Muhammad Ilyas. 2022. Systematic Literature Review on Security Risks and its Practices in Secure Software Development. *IEEE Access* 10 (01 2022), 5456–5481. https://doi.org/10.1109/ACCESS.2022.3140181

[9] Rafiq Ahmad Khan, Siffat Ullah Khan, Habib Ullah Khan, and Muhammad Ilyas. 2021. Systematic Mapping Study on Security Approaches in Secure Software Engineering. *IEEE Access* 9 (2021), 19139–19160. https://doi.org/10.1109/ACCESS.2021.3052311

[10] Torsten Lodderstedt, David Basin, and Jürgen Doser. 2002. *SecureUML: A UML-Based Modeling Language for Model-Driven Security.* 426–441. https://doi.org/10.1007/3-540-45800-X_33

[11] Zulfikar Maher, Mansoor Hyder, M Memon, Dr. Muhammad Yaqoob Koondhar, Pinial Khan, and Advsunil Shah. 2020. Measuring Secure Software Development Awareness and Usage among Software Developers. (12 2020). https://doi.org/10.26692/sujo/2020.12.57

[12] Fabio Massacci, John Mylopoulos, and Nicola Zannone. 2007. Computer-aided Support for Secure Tropos. *Automated Software Engineering* 14 (9 2007). https://doi.org/10.1007/s10515-007-0013-5

[13] John Mcdermott and Chris Fox. [n. d.]. *Using Abuse Case Models for Security Requirements Analysis.* Technical Report.

[14] G. McGraw. 2004. Software Security. *IEEE Security and Privacy* 2 (2004), 80–83. Issue 2. https://doi.org/10.1109/MSECP.2004.1281254

[15] Nancy R. Mead and Ted Stehney. 2005. Security quality requirements engineering (SQUARE) methodology. *ACM SIGSOFT Software Engineering Notes* 30 (7 2005), 1–7. Issue 4. https://doi.org/10.1145/1082983.1083214

[16] Daniel Mellado, Carlos Blanco, Luis E. Sánchez, and Eduardo Fernández-Medina. 2010. A systematic review of security requirements engineering. *Computer Standards & Interfaces* 32, 4 (2010), 153–165. https://doi.org/10.1016/j.csi.2010.01.006

[17] Anuradha Misra. 2017. Aspects of Enhancing Security in Software Development Life Cycle. *International Journal for Computational Methods in Engineering Science and Mechanics* 1 (01 2017), 203–210.

[18] Anuradha Misra. 2017. Aspects of Enhancing Security in Software Development Life Cycle. *International Journal for Computational Methods in Engineering Science and Mechanics* 1 (01 2017), 203–210.

[19] Malik Nadeem Anwar, Mohammed Nazir, and Prof K. Mustafa. 2019. A Systematic Review and Analytical Evaluation of Security Requirements Engineering Approaches. *Arabian Journal for Science and Engineering* 44 (08 2019).

https://doi.org/10.1007/s13369-019-04067-3

[20] Hernan Nina, Jose Antonio Pow-Sang, and Monica Villavicencio. 2021. Systematic Mapping of the Literature on Secure Software Development. *IEEE Access* 9 (3 2021), 36852–36867. https://doi.org/10.1109/ACCESS.2021.3062388

[21] P. Salini and S. Kanmani. 2012. Survey and analysis on Security Requirements Engineering. *Computers & Electrical Engineering* 38 (11 2012), 1785–1797. Issue 6. https://doi.org/10.1016/j.compeleceng.2012.08.008

APÊNDICE B - Protocolo

Case Study Planning

TITLE: Towards a Security Requirement Specification Approach for Health Systems in Agile Contexts: a Case Study

THEME: Study a method to support the derivation of security requirements to simplify and improve the understanding of security

AUTHORS: Edilson do Nascimento Costa Júnior

Case Study Definition

Research objective: Study the SecurityRE approach with the aim of specifying security requirements in terms of the suitability of the method in agile development, the compliance of the generated artifacts with security standards, and the practicality of use for developers with little connection to security.

Context: The case study was carried out with computer scientists and students in the context of agile development of healthcare systems.

Research questions:

- **Suitability**: How well does the method for deriving security requirements meet the specific requirements in the agile environment in which it is implemented?
- **Completeness/Compliance:** To what extent does the security requirements derivation method generate artifacts that comply with security standards and regulations such as HIPAA, GDPR, and ISO/IEC 27001?
- **Usefulness**: How clear, simple and easy is the method of deriving security requirements for agile developers who have little or no connection to security?

Data Collection Procedure(s):

Surveys:

- Distribute surveys to a larger group of developers and students to gather quantitative data on their perceptions of the method's suitability, and usefulness.
- The survey questions are asked on the Likert scale.

Document Analysis:

- Review artifacts generated by the method, such as security requirement documents, and security stories.
- Analyzes artifacts to obtain evidence of compliance and completeness with HIPAA, GDPR, ISO/IEC 27001 and other relevant standards.

Analysis Procedure(s)

Qualitative Analysis:

- Compare the artifacts generated by the method against the requirements of HIPAA, GDPR, ISO/IEC 27001, NIST SP 800-53, and other relevant standards.
- Assess the integrity and accuracy of artifacts and look for evidence of compliance.

Quantitative Analysis:

- Analyze survey data using descriptive and inferential statistics to identify patterns and correlations in participants' perceptions of the method's suitability and usefulness.

VARIABLES SELECTION:

- **Independent Variables:**
  - Security requirement derivation method used (SecurityRE approach);
  - Subjects experience with security (none, low, medium, high);

- **Dependent Variables:**
  - Method suitability (measured by participants' perceptions in interviews and surveys);
  - Completeness/Compliance of generated artifacts (measured by document analysis against security standards);
  - Method usefulness (measured by the clarity, simplicity, and ease of use perceived by participants);

Metrics:

1. Suitability:

- Measured through Likert scale-based survey questions.
- Sample questions include:
  - "As histórias de detecção do método são eficientes para você identificar possíveis violações de segurança."
  - "As histórias de prevenção fornecem diretrizes práticas e aplicáveis para você."
  - "As histórias de mitigação são adequadas para você evitar ataques e vulnerabilidades comuns."

2. Compliance:

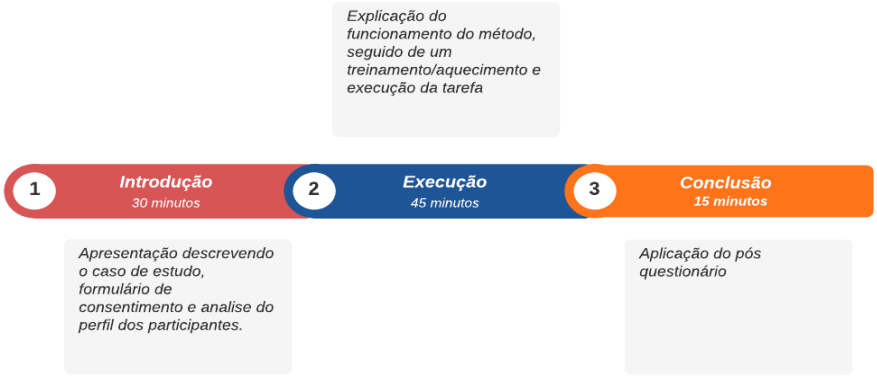
- Measured by comparing the generated artifacts against established standards such as HIPAA, GDPR, and NIST.
- Key areas of comparison include:
  - Access control
  - Data Integrity and Confidentiality
  - Encryption

3. Usefulness:

- Measured through Likert scale-based survey questions and interviews.
- Sample questions include:
    - "O método de derivação de requisitos de segurança é claro para você, mesmo tendo pouca ou nenhuma conexão com segurança."
    - "O método de derivação de requisitos de segurança é simples de entender e aplicar para você."
    - "O método de derivação de requisitos de segurança é fácil de seguir sem que você precise de conhecimento especializado em segurança."

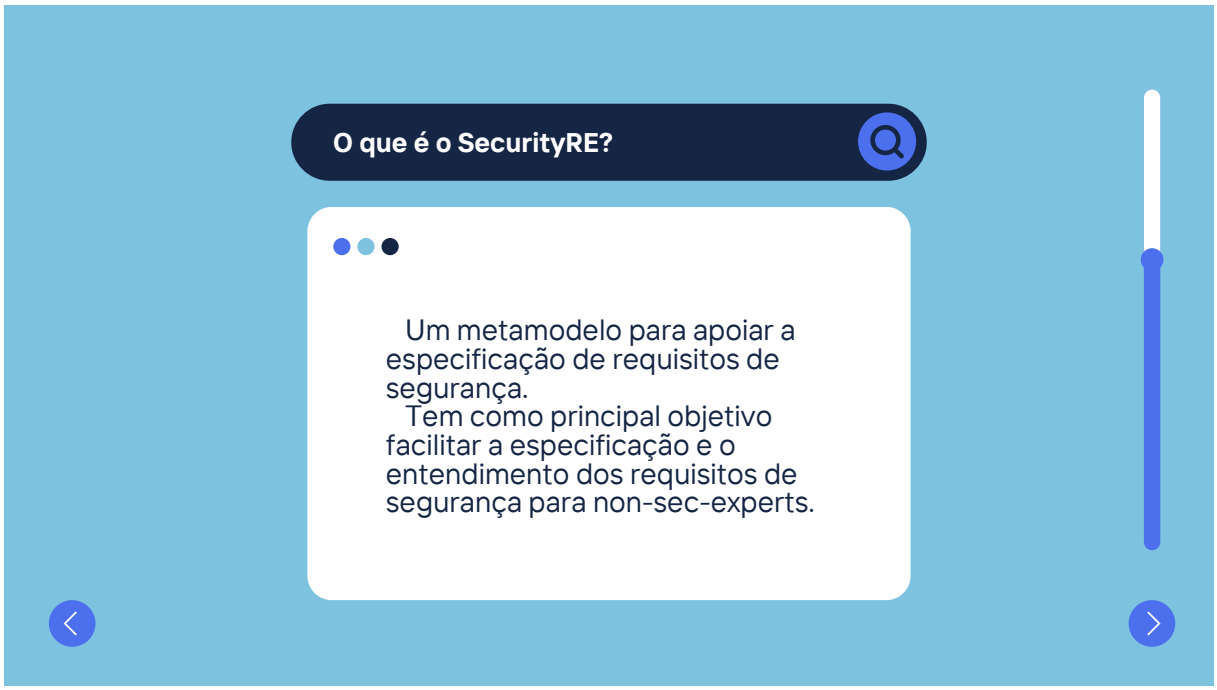
Conducting Procedure:

1. Project Overview(objetivo, introdução ao método) - 15 minutes
2. Informed consent
3. Survey (subjects perfil) - 5 minutes
4. Method - 15 minutes
5. Training (warm up) - 20 minutes
6. Task - 20 minutes
7. Post questionnaire (feedback) - 15 minutes



APÊNDICE C - Apresentação caso de estudo


Requisitos de segurança


O que é o SecurityRE?
Um metamodelo para apoiar a especificação de requisitos de segurança.
Tem como principal objetivo facilitar a especificação e o entendimento dos requisitos de segurança para non-sec-experts.

Estrutura do método

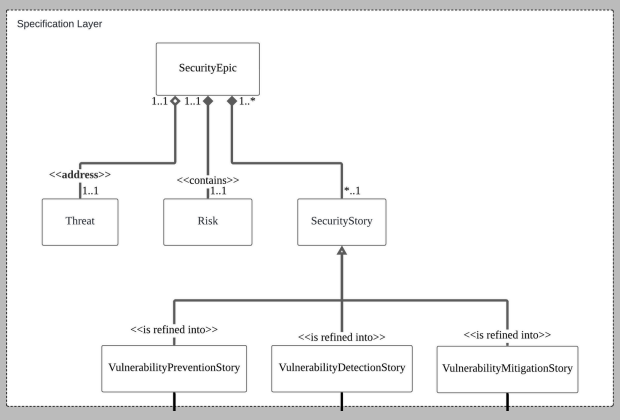
1. **Threat**
   As possíveis ameaças são identificadas e listadas.
2. **Security Epic**
   É escrita para cada ameaça encontrada.
3. **Security story**
   Descreve de forma funcional a SE.
4. **Risk**
   Um risco é definido e associado a cada SE.

Estrutura do método

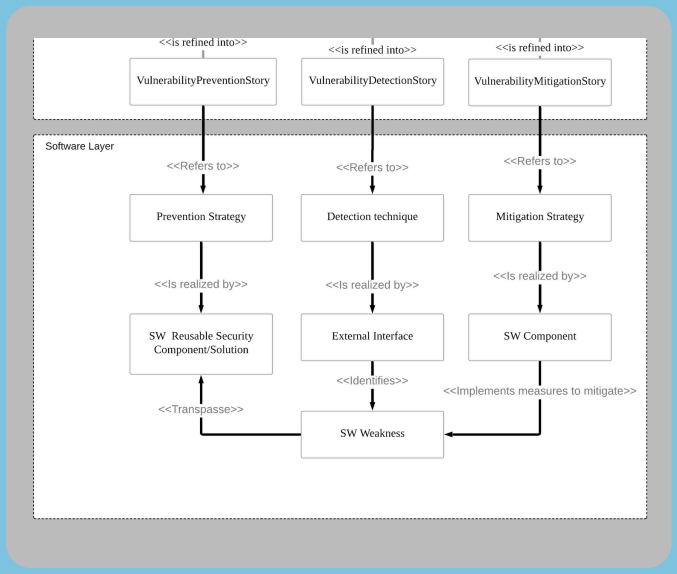
1. **Vulnerability Prevention Story**
   Descreve um método de PREVENÇÃO para a vulnerabilidade associada.

2. **Vulnerability Detection Story**
   Descreve um método de DETECÇÃO para a vulnerabilidade associada.

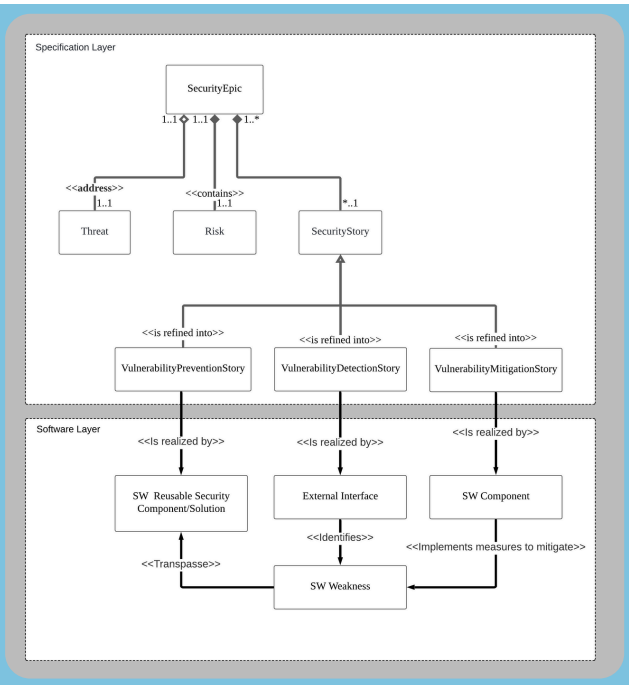
3. **Vulnerability Mitigation Story**
   Descreve um método de MITIGAÇÃO para a vulnerabilidade associada.



Estrutura do método

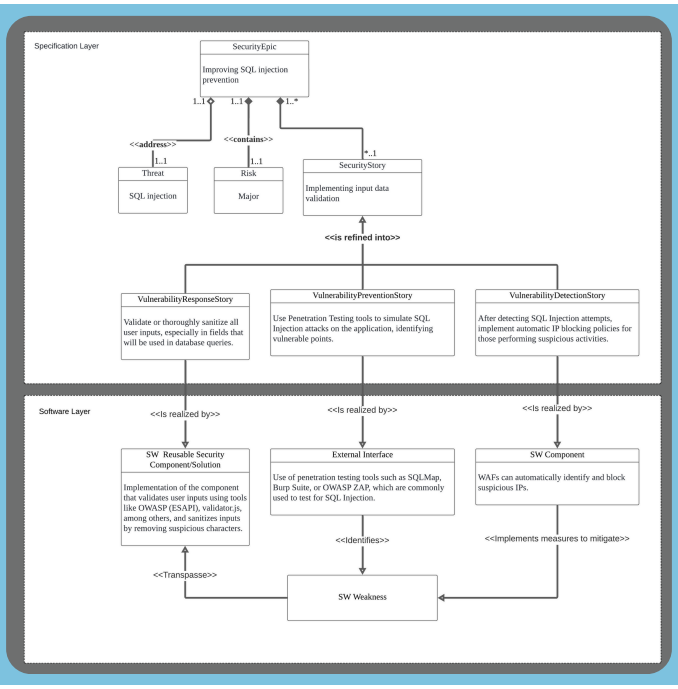


Estrutura do método



Estrutura do método

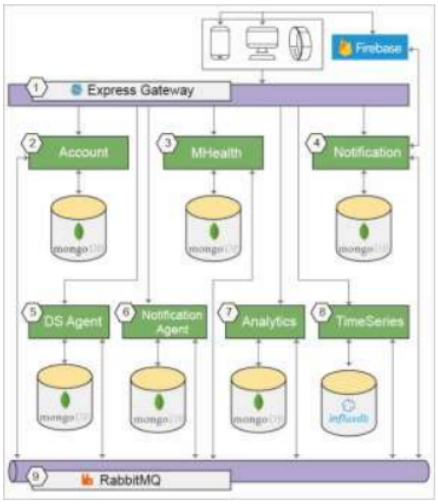
Exemplo do método



Vamos ao treino

Sistema de acompanhamento:

A plataforma Senior Mobile Health (SMH), desenvolvida para o monitoramento remoto de idosos utilizando dispositivos vestíveis (wearables) que coletam dados de saúde, com foco na variabilidade da frequência cardíaca (HRV). A plataforma integra-se com dispositivos de IoT, como smartwatches, e é destinada a auxiliar profissionais de saúde, cuidadores e os próprios idosos no gerenciamento de diversos indicadores de saúde, como frequência cardíaca, atividade física, padrões de sono, fragilidade e risco de quedas.


Firebase
1
Express Gateway
2
Account
3
MHealth
4
Notification
5
DS Agent
6
Notification Agent
7
Analytics
8
TimeSeries
9
RabbitMQ

Requisitos do sistema

- O sistema deve permitir que administradores, profissionais de saúde, cuidadores e pacientes realizem login por meio de suas contas.
- O sistema deve permitir a sincronização automática de dados entre dispositivos vestíveis (smartwatches) e smartphones.
- O sistema deve possibilitar que os profissionais de saúde registrem e consultem dados médicos dos pacientes em tempo real.
- O sistema deve permitir a criação de metas personalizadas para os pacientes, como quantidade diária de passos.
- O sistema deve permitir que os pacientes visualizem seus dados de saúde, como passos, frequência cardíaca e qualidade do sono, no aplicativo móvel.
- O sistema deve calcular automaticamente indicadores de saúde como velocidade da marcha e frequência cardíaca, e disponibilizá-los para os profissionais de saúde.
- O sistema deve permitir que os profissionais de saúde pesquisem e acompanhem o histórico médico dos pacientes.

Objetivos de segurança

- O sistema deve ser protegido contra SQL Injection.
- O sistema deve proteger a comunicação entre o servidor e os dispositivos (smartphones, wearables) contra ataques do tipo Man-in-the-Middle (MITM).
- O sistema não deve permitir ataques de força bruta.
- O sistema deve garantir a segurança de todos os dados sensíveis, como informações médicas.
- O sistema deve proteger contra Cross-Site Request Forgery (CSRF).
- O sistema deve proteger contra Cross-Site Scripting (XSS).
- O sistema deve garantir que todas as APIs expostas sejam protegidas com autenticação e autorização adequadas.
- O sistema deve prevenir contra privilege escalation.

Micro-serviços

01 **Express Gateway:**

Gerencia as requisições da API, funcionando como proxy entre os clientes externos e os serviços internos.

02 **Serviço de Conta:**

Responsável pela autenticação e armazenamento das informações dos usuários.

03 **MHealth:**

Armazena medições de saúde, como HRV, peso, pressão arterial, atividade física, entre outros.

04 **Serviço de Notificações**

Gerencia o envio de e-mails, SMS e notificações push usando o Firebase.

Micro-serviços

05 **Agente de Sincronização de Dados (DS Agent):**

Sincroniza os dados dos dispositivos vestíveis com a plataforma SMH, pré-processando os dados para adequá-los ao modelo da plataforma.

07 **Séries Temporais:**

Armazena dados de séries temporais (como frequência cardíaca, passos).

06 **Analytics:**

Processa e analisa os dados de saúde para fornecer insights e informações relevantes.

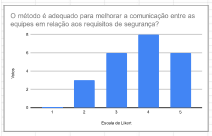
08 **RabbitMQ Message Channel:**

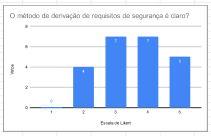
Facilita a comunicação assíncrona entre os microserviços.

Vamos ao treino

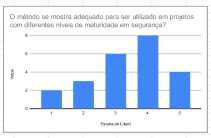
Utilize frameworks de segurança como Mitre Att&ck, OWASP, NIST, NVD e entre outros para ajudar na derivação dos requisitos de segurança ao utilizar o método.

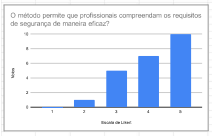
APÊNDICE D - Resultados Formulário sobre Usefulness


O método é adequado para melhorar a comunicação entre as equipes em relação aos requisitos de segurança?
Votos
Escala de Likert

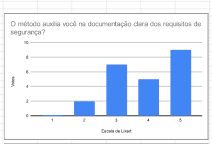

O método de derivação de requisitos de segurança é claro?
Votos
Escala de Likert

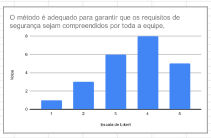

Você considera que o método tem uma complexidade alta?
Votos
Escala de Likert


O método se mostra adequado para ser utilizado em projetos com diferentes níveis de maturidade em segurança?
Votos
Escala de Likert

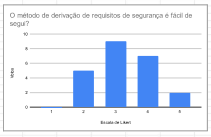

O método permite que profissionais compreendam os requisitos de segurança de maneira eficaz?
Votos
Escala de Likert


O método de derivação de requisitos de segurança é simples de entender e aplicar para você.
Votos
Escala de Likert

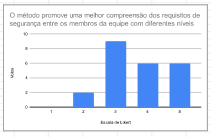

O método auxilia você na documentação clara dos requisitos de segurança?
Votos
Escala de Likert


O método é adequado para garantir que os requisitos de segurança sejam compreendidos por toda a equipe.
Votos
Escala de Likert

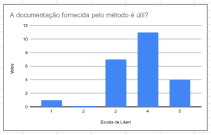

O método se integra bem às cerimônias e práticas ágeis, como sprints e reuniões diárias.
Votos
Escala de Likert


O método de derivação de requisitos de segurança é fácil de seguir?
Votos
Escala de Likert

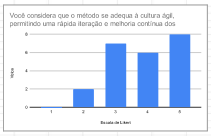

O método é útil para desenvolvedores com diferentes níveis de experiência em segurança?
Votos
Escala de Likert


O método promove uma melhor compreensão dos requisitos de segurança entre os membros da equipe com diferentes níveis
Votos
Escala de Likert

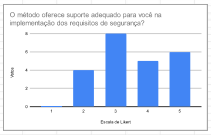

O método promove um melhor entendimento dos riscos de segurança entre os membros da equipe com pouca experiência
Votos
Escala de Likert


A documentação fornecida pelo método é útil?
Votos
Escala de Likert


O método se adapta bem ao ambiente de desenvolvimento ágil.
Votos
Escala de Likert


Você considera que o método se adequa à cultura ágil, permitindo uma rápida iteração e melhoria contínua dos
Votos
Escala de Likert


O método te auxilia a reduzir a complexidade percebida por você na implementação de requisitos de segurança?
Votos
Escala de Likert


O método oferece suporte adequado para você na implementação dos requisitos de segurança?
Votos
Escala de Likert

APÊNDICE E - Questionário sobre o perfil do participante

Carimbo de data/hora	Qual é o seu nível de escolaridade?	Quantos anos de experiência você tem no desenvo	Você tem conhecimento/ experiência em Segurança	Como você classifica sua experiência em relação a	Quantos anos de experiência você tem utilizando m	Como você classifica seu conhecimento em relação	Quão familiarizado você está com as normas de se	Quão confortável você se sente ao especificar requ
	Estudante de graduação	Menos de 1 ano	Não	Regular - Participei de um projeto acadêmico com i	1-3 anos	Regular - Tenho conhecimento das normas, embora	Nunca ouvi falar	Extremamente confortável
	Estudante de graduação	1-3 anos	Sim	Ótima - Participei de vários projetos em empresas i	1-3 anos	Bom - Participei de projetos nos quais foram espec	Conheço superficialmente	Confortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Extremamente desconfortável
	Estudante de graduação	Nenhum	Não	Pouca - Apenas em projetos de disciplinas na unive	Nenhum	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Extremamente desconfortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	1-3 anos	Pouco - Apenas em disciplinas na universidade	Nunca ouvi falar	Moderadamente confortável
	Estudante de graduação	Menos de 1 ano	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Nunca ouvi falar	Extremamente desconfortável
	Estudante de graduação	Nenhum	Não	Nenhum - Nunca desenvolvi ou participei de projeto	Nenhum	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Desconfortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Nunca ouvi falar	Moderadamente confortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Regular - Tenho conhecimento das normas, embora	Conheço superficialmente	Desconfortável
	Estudante de graduação	1-3 anos	Sim	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Regular - Tenho conhecimento das normas, embora	Conheço superficialmente	Desconfortável
	Estudante de graduação	1-3 anos	Não	Boa - Participei de um projeto em uma empresa qu	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Extremamente desconfortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Nenhum - Nunca estudei, não sei do que se trata	Conheço superficialmente	Desconfortável
	Bacharel	Menos de 1 ano	Sim	Pouca - Apenas em projetos de disciplinas na unive	1-3 anos	Regular - Tenho conhecimento das normas, embora	Nunca ouvi falar	Desconfortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	1-3 anos	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Moderadamente confortável
	Estudante de graduação	1-3 anos	Sim	Regular - Participei de um projeto acadêmico com i	Menos de 1 ano	Bom - Participei de projetos nos quais foram espec	Tenho conhecimento básico	Confortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	1-3 anos	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Desconfortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	1-3 anos	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Extremamente desconfortável
	Estudante de graduação	1-3 anos	Sim	Boa - Participei de um projeto em uma empresa qu	1-3 anos	Bom - Participei de projetos nos quais foram espec	Conheço superficialmente	Extremamente desconfortável
	Estudante de graduação	1-3 anos	Sim	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Bom - Participei de projetos nos quais foram espec	Conheço superficialmente	Moderadamente confortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Nenhum	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Desconfortável
	Estudante de graduação	Menos de 1 ano	Sim	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Desconfortável
	Estudante de graduação	4-6 anos	Não	Ótima - Participei de vários projetos em empresas i	Menos de 1 ano	Bom - Participei de projetos nos quais foram espec	Conheço superficialmente	Moderadamente confortável
	Estudante de graduação	1-3 anos	Não	Pouca - Apenas em projetos de disciplinas na unive	Menos de 1 ano	Pouco - Apenas em disciplinas na universidade	Conheço superficialmente	Extremamente desconfortável