doi.org/10.51891/rease.v10i3.13166

PRÁTICAS DE SEGURANÇA NO DESENVOLVIMENTO COM O USO DE METODOLOGIAS ÁGEIS

Ialle Teixeira da Conceição[1]

RESUMO: No início dos anos 1990, com o crescimento do uso de computadores por parte das empresas, o desenvolvimento de software enfrentou uma crise com muitos atrasos nas entregas de aplicativos. Dessa forma, diversos profissionais começaram a trabalhar em busca de novas abordagens para eliminar o que consideravam atividades improdutivas dentro do desenvolvimento de software e gerar modelos que se adaptassem melhor à modernidade. Nesse contexto, surgiram os diversos tipos de metodologias ágeis. O foco desses métodos tem sido dinamizar os processos de desenvolvimento de software de modo a eliminar os principais problemas que as empresas enfrentam em projetos. São estratégias e técnicas direcionadas para a colaboração de todos os membros dentro de pequenos ciclos de trabalho, utilizando o desenvolvimento incremental. Ao lançar mão dos processos ágeis, é possível otimizar a eficiência durante a execução, pois uma grande tarefa se divide em etapas menores. Assim, são obtidos melhores e mais rápidos resultados, já que se cria um ambiente de constante diálogo. Isso permite o controle de recursos e a transparência, além de garantir o feedback do cliente em cada etapa. Tais ações tornam o manejo das mudanças mais simples do que ao final de todos os processos. O presente trabalho bibliográfico fez uma abordagem sobre as práticas de segurança no desenvolvimento com o uso de metodologias ágeis.

Palavras-chave: Práticas de segurança. Metodologias ágeis. Segurança de softwares.

ABSTRACT: In the early 1990s, with the growth in the use of computers by companies, software development faced a crisis with many delays in application deliveries. In this way, several professionals began to work in search of new approaches to eliminate what they considered unproductive activities within software development and generate models that better adapted to modernity. In this context, different types of agile methodologies emerged. The focus of these methods has been to streamline software development processes in order to eliminate the main problems that companies face in projects. These are strategies and techniques aimed at the collaboration of all members within small work cycles, using incremental development. By using agile processes, it is possible to optimize efficiency during execution, as a large task is divided into smaller steps. This way, better and faster results are obtained, as an environment of constant dialogue is created. This allows for resource control and transparency, as well as ensuring customer feedback at every stage. Such actions make managing changes simpler than at the end of all processes. This bibliographical work approached security practices in development using agile methodologies.

Keywords: Security practices. Agile methodologies. Software Security.

[1]Pós-graduando em Engenharia de Software pela Universidade Estácio de Sá (UNESA).

INTRODUÇÃO

Hoje em dia, as empresas são apoiadas na maior parte das suas tarefas graças à tecnologia, desde as comunicações entre diferentes sistemas de negócio para a entrega de um resultado ou funcionalidade até à gestão interna da informação, tendo em conta que esta abrange utilizações muito amplas dependendo da abordagem gerida pela empresa, o que significa que existem grandes riscos em cada uma das aplicações tecnológicas.

Além disso, estes riscos evoluem com a tendência de tecnologias como a computação em nuvem ou aplicações empresariais concebidas para dispositivos móveis. A nível nacional, os ataques informáticos impactam as empresas com perdas imensuráveis, no mundo os números são ainda maiores, já que boa parte das empresas no mundo relataram algum tipo de ataque cibernético.

Um cenário que não pode passar despercebido às pessoas ligadas ao desenvolvimento de software que devem começar a desenhar a utilização de metodologias que garantam o lugar que corresponde à segurança sem deixar de lado a agilidade e confiabilidade com que estas são desenvolvidas.

Devido ao aumento diário de ataques informáticos a diversas entidades e ao risco em termos de dinheiro e informação que isso representa para as empresas, deve ser considerada uma medida eficaz que permita adaptar os projetos de desenvolvimento para ter em conta o desenvolvimento de software seguro entre as suas prioridades, sem impactar muito na sua execução em termos de tempo, agilidade e qualidade.

Portanto, torna-se necessário estimar como as metodologias ágeis utilizadas nas empresas podem afetar o desenvolvimento seguro de software, para que elas levem isso em consideração na hora de decidir qual metodologia utilizar no projeto de forma eficaz ao mesmo tempo que se faz necessário avaliar como o desenvolvimento seguro pode ser eficiente e não disruptivo em equipes ágeis.

Diante de tal questão, tem-se a seguinte situação-problema: Quais são as implicações de maior impacto no desenvolvimento de software seguro com metodologias ágeis nas empresas?

Uma das necessidades mais importantes nas empresas é obter uma estimativa real do dinheiro e do tempo que custará para executar um determinado projeto de software. Se a importância da segurança na aplicação for incluída adicionalmente, é muito provável que os

resultados da estimativa estarão errados, devido à falta de conhecimento sobre qual metodologia utilizar nas empresas de acordo com o escopo do projeto de software.

Adicionalmente, a construção de software seguro contribui para o modelo de continuidade de negócio sistema de gestão de segurança da informação na medida em que permite identificar e analisar as vulnerabilidades que possam existir no software para posteriormente desenhar estratégias de recuperação e estabelecer procedimentos correspondentes à moderação de ataques no sistema.

É necessário um estudo para analisar a forma como este tipo de projetos estão sendo realizados, se utilizam metodologia ágil ou tradicional, qual a percepção da importância da segurança para o software construído e quais as estratégias contempladas para mitigar ataques a as aplicações desenvolvidas, dada a tendência de aumento dos ataques informáticos que têm ocorrido nos últimos anos em todo o mundo e a importância da informação que pode ser extraída em cada ataque, bem como a funcionalidade cumprida pelo sistema atacado.

Objetivo geral será determinar as principais implicações de segurança nas metodologias ágeis de desenvolvimento de software no contexto nacional. Objetivos específicos são: apresentar como ocorrem os projetos de software seguros em e empresas de tecnologia quando utilizam metodologias ágeis; concluir as consequências da integração da construção de software seguro às metodologias ágeis; e avaliar como o desenvolvimento seguro pode ser eficiente e não disruptivo em equipes ágeis.

Espera-se que os resultados obtidos neste estudo forneçam um contexto claro de como as empresas estão no uso de boas práticas para realizar projetos de software seguros com metodologias ágeis de forma correta e eficiente.

Fundamentação teórica

Segurança de Softwares

Segundo Souza (2023) a segurança de software é a aplicação de princípios de segurança da informação ao desenvolvimento de software. Por sua vez, a segurança da informação refere-se à proteção dos sistemas de informação contra o acesso não autorizado ou modificação do mesmo, se estiver em armazenamento, processamento ou em estágio de trânsito.

Para Mendoza e Pizzolato (2023) a segurança de software é uma ideia da Engenharia de Software que busca garantir que um produto desenvolvido continue funcionando corretamente diante de ataques maliciosos, é a construção de um software que possa resistir proativamente aos ataques.

Além de um aspecto crítico dos problemas de segurança estão os problemas apresentados pelo próprio software. Souza (2023) enfatiza que a segurança não é um aspecto que possa ser incorporado a um sistema a qualquer momento, o que implica que, ao projetar um produto de software, esse design deve ser realizado tendo em mente o aspecto da segurança, visto que, uma vez desenvolvido o produto, não será possível incorporá-lo com sucesso.

Interpretando o exposto, Felipe (2021) propõe métodos que podem ser adaptados à forma como o software for desenvolvido, entre os mais relevantes estão: a revisão de código utilizando ferramentas de análise estática; a análise de risco arquitetônico; os testes de segurança; e o desenvolvimento de casos de abuso.

A segurança no ciclo de vida de software é a adoção de um modelo encarregado de abordar a segurança para cada uma das fases do ciclo de vida, uma das referências mais adequadas. Normalmente, o tipo de software de segurança mais conhecido é o antivírus, porém, existem outros componentes de segurança responsáveis por complementar a função do antivírus, alguns deles são: programas antivírus; firewall; filtro de spam; softwares de filtragem de conteúdo; software antipublicidade; e controle do site. (SOUZA, 2023).

Definido por Mendoza e Pizzolato (2023), nesta fase devem ser identificados os requisitos funcionais que terão impacto nos aspectos de segurança da aplicação. Alguns deles são: requisitos de conformidade com regulamentações locais ou internacionais, tipo de informação que será transmitida ou processada e requisitos de registro de auditoria.

Uma vez concluído o design, inicia-se a fase de desenvolvimento com a codificação dos diferentes componentes da aplicação. É neste ponto que normalmente são incorporados diferentes tipos de vulnerabilidades, por erro ou omissão. (SOUZA, 2023).

Os 'Frameworks' de desenvolvimento de aplicações são uma boa ajuda neste ponto, pois atuam como intermediários entre o programador e o código, e permitem que a maioria das vulnerabilidades conhecidas sejam evitadas. Souza (2015) divide essas vulnerabilidades em dois grandes grupos.

Vulnerabilidades Clássicas, um exemplo dessas vulnerabilidades são buffer overflows, negação de serviço, entre outras. Os 'Frameworks' de desenvolvimento de aplicações são uma boa ajuda neste ponto, pois atuam como intermediários entre o programador e o código, e permitem que a maioria das vulnerabilidades conhecidas sejam evitadas. (SOUZA, 2023).

Conforme apreciado por Mendoza e Pizzolato (2023) as vulnerabilidades funcionais, são aqueles especificamente ligados à funcionalidade de negócio que a aplicação possui, portanto não são previamente categorizados e dependem diretamente das informações ou serviços utilizados pelo negócio.

Alguns exemplos deste tipo de vulnerabilidade são os seguintes: uma aplicação de banca electrónica que permite transferências com valores negativos, um sistema de leilão que permite ver os valores de outros licitantes, um sistema de venda de bilhetes para espetáculos que não impõe limites adequados ao número de reservas que um usuário pode fazer. (SOUZA, 2023).

Tudo o que foi descrito acima é feito para mitigar incidentes de segurança naquilo que podemos referir: acesso não autorizado à informação; descoberta de informações; modificação não autorizada de dados; invasão de privacidade; negação de serviços. (FELIPE, 2021).

Conforme proposto em Souza (2023) quando um malware é descoberto em um sistema, há muitas decisões e ações que devem ser tomadas, muitas vezes sob forte pressão de tempo. Para ajudar os investigadores digitais a alcançar um resultado bem sucedido, uma metodologia geral pode ser usada para lidar com estes incidentes, investigações envolvendo malware em cinco fases.

A primeira se refere a preservação forense e exame de dados voláteis. O valor dos dados voláteis não se limita à memória do processo associada ao malware, mas pode incluir senhas, endereços de protocolo de Internet (IP), entradas de log de eventos de segurança e outros detalhes contextuais que podem fornecer informações adicionais em um sistema. (MENDOZA; PIZZOLATO, 2023).

Em um estado ligado, um sistema em questão contém informações efêmeras críticas que revelam o estado do sistema. Esses dados voláteis são às vezes chamados de informações com estado. A resposta forense a incidentes, ou resposta ao vivo, é o processo de aquisição

das informações de status do sistema em questão enquanto ele permanece ligado. (SOUZA, 2023)

O segundo se refere ao teste de memória. Após adquirir uma imagem de memória física, é necessário extrair informações, conteúdo de forma metódica. Uma captura de memória completa pode conter evidências críticas em um incidente de código malicioso, incluindo quando o malware foi iniciado, argumentos de linha de comando usados, processos ocultos e encerrados, endereços IP com os quais o malware se comunicou e dados em texto cifrado no disco.

Algumas ferramentas forenses de memória podem listar arquivos abertos, conexões de rede ativas e processos em execução, e podem até exibir informações sobre processos que estão ocultos ou que não estão mais em execução, mas que ainda estão presentes na memória. (SOUZA, 2023).

Embora os pesquisadores digitais muitas vezes encontrem informações úteis em despejos de memória simplesmente revisando texto legível e realizando pesquisas por palavras-chave. Contexto e metadados adicionais só podem ser obtidos usando conhecimento especializado das estruturas de dados na memória. (MENDOZA; PIZZOLATO, 2023).

A localização de dados associados a um processo específico é complicada pelo fato de os sistemas operacionais Windows e Linux usarem endereços de sistema virtuais para criar a ilusão de mais memória do que a existente fisicamente. (SOUZA, 2023).

Como resultado, para encontrar um determinado dado, é necessário traduzir endereços virtuais em um local físico. Além disso, a localização física dos dados pode estar em um arquivo de paginação no disco, em vez de no despejo de memória física. (FELIPE, 2021).

O terceiro se refere ao exame de análise forense de discos rígidos. O exame forense de sistemas Windows é uma parte importante da análise de códigos maliciosos, fornecendo contexto e informações adicionais que nos ajudam a compreender a funcionalidade e a origem do malware. (SOUZA, 2015).

Na medida em que a análise de sistemas ligados pode ser considerada uma cirurgia, o exame forense pode ser considerado uma autópsia de um computador afetado por malware. (SOUZA, 2023). Evidências de rastreamento relacionadas a um determinado malware podem ser encontradas em sistemas operacionais e no sistema de arquivos, incluindo

arquivos, entradas de registro, registros em logs de eventos e carimbos de data associados. (MENDOZA; PIZZOLATO, 2023).

O quarto se refere a análise estática de malware. São as técnicas e ferramentas para realizar uma análise inicial de um arquivo suspeito com o objetivo de analisar o código assembly do malware para obter melhor compactação e funcionamento. (SOUZA, 2023).

A quinta se refere a análise dinâmica de malware. A análise dinâmica ou comportamental envolve a execução do código e o monitoramento de seu comportamento, interação e efeito no sistema host, enquanto a análise estática é o processo de análise do código binário executável sem realmente executar o arquivo. (FELIPE, 2021).

Conforme (SOUZA, 2023) Buffer Overflows são um tipo de vulnerabilidade de uso de memória. Isso é principalmente um acidente da história da ciência da computação. A memória era um recurso precioso, portanto, gerenciá-la era fundamental.

Em alguns sistemas mais antigos, como a espaçonave Voyager, a memória era tão valiosa que, quando certas seções do código de máquina não eram mais necessárias, o código era apagado para sempre do módulo de memória, liberando espaço para outros usos. (MENDOZA; PIZZOLATO, 2023).

Isso efetivamente criou um programa autodestrutivo e que só poderia ser executado uma vez. A maioria dos sistemas de softwares conectados em rede hoje apresenta problemas de memória abomináveis, especialmente quando conectados diretamente a ambientes hostis como a Internet. (SOUZA, 2023).

A memória é barata, mas os efeitos do mau gerenciamento da memória são muito caros. O uso indevido de memória pode levar à corrupção interna de um programa (especialmente com referência a um fluxo de controle) e a problemas de negação de serviço.

De acordo com Souza (2023) os testes de segurança representam um problema único. A maioria dos defeitos e vulnerabilidades de segurança em software não está diretamente relacionada à funcionalidade. Em vez disso, as questões de segurança envolvem abusos inesperados, mas intencionais, de um aplicativo descoberto por um invasor.

Se caracterizarmos o teste funcional como "teste positivo" como verificação de que um recurso executa adequadamente uma tarefa específica, então o teste de penetração é, em certo sentido, "teste negativo", ou seja, um testador de segurança deve investigar direta e profundamente a segurança (possivelmente motivada por casos de abuso e riscos

arquitetônicos) para determinar como o sistema sob ataque se comporta. (MENDOZA; PIZZOLATO, 2023).

O teste de penetração envolve testar um sistema em seu ambiente de produção final. Por esse motivo, os testes de penetração são mais adequados para testar problemas de configuração e outros fatores ambientais que afetam profundamente a segurança do software. (SOUZA, 2023).

Os testes de condução que se concentram nestes fatores com algum conhecimento dos resultados da análise de risco são a abordagem mais eficaz. Não se deve esquecer que o valor real dos testes de penetração advém do seu papel central na validação da configuração e de outros fatores ambientais essenciais. (FELIPE, 2021).

Como ferramenta de medição, o teste de penetração é mais poderoso quando é totalmente integrado ao processo de desenvolvimento, de tal forma que os resultados iniciais do ciclo de vida são usados para informar os testes e os resultados retroalimentam as práticas de desenvolvimento e implantação. (SOUZA, 2023).

Conforme Mendoza e Pizzolato (2023) os testes de segurança devem abranger duas estratégias: testar a funcionalidade de segurança com técnicas de teste funcional padrão e testes de segurança baseados em riscos, baseados em padrões de segurança.

Um bom plano de testes de segurança abrange ambas as estratégias. Os problemas de segurança nem sempre são óbvios, mesmo quando um sistema é examinado diretamente, por isso é pouco provável que a garantia de qualidade dos problemas padrão revele todos os problemas críticos de segurança. (SOUZA, 2023).

Garantia de qualidade é garantir que coisas boas aconteçam. Os testes de segurança visam garantir que coisas ruins não aconteçam. Pensar como um invasor é essencial. Orientar os testes de segurança com conhecimento da arquitetura de software, ataques comuns e mentalidade do invasor é, portanto, extremamente importante. (SOUZA, 2015).

Os testadores devem adotar uma abordagem baseada em riscos, baseada na realidade arquitetônica do sistema e na mentalidade do invasor, para medir adequadamente a segurança do software. Ao identificar riscos no sistema e criar testes orientados por esses riscos, um testador de segurança de software pode se concentrar adequadamente nas áreas do código onde um ataque provavelmente terá sucesso. (SOUZA, 2023).

Essa abordagem fornece um nível mais alto de garantia de segurança de software do que é possível com testes clássicos de caixa preta. Os profissionais de segurança de software

executam muitas tarefas diferentes para gerenciar riscos de segurança de software, como: criar casos de uso indevido de segurança; listar requisitos regulamentares de segurança; realizar análise de risco arquitetural; criar planos de testes de segurança baseados em risco; utilizar ferramentas de análise estática; realizar testes de segurança; realizar testes de penetração no ambiente final; e limpeza após violações de segurança. (MENDOZA; PIZZOLATO, 2023).

Os testes de segurança devem necessariamente incluir duas abordagens diferentes: testes de segurança funcional, mecanismos de segurança para garantir que a sua funcionalidade é implementada corretamente; e testes de segurança adversos que realize testes de segurança baseados em riscos, motivados pela compreensão e simulação da abordagem do invasor. (SOUZA, 2023).

METODOLOGIAS ÁGEIS

As práticas ágeis utilizadas na indústria para desenvolvimento de software e a busca pelo comprometimento com qualidade, escopo e tempo no desenvolvimento de projetos pode ser realizada por meio de práticas de desenvolvimento e teorias de gestão desenhadas para esse fim, como metodologias ágeis, por isso práticas de desenvolvimento de software são um conjunto de atividades ou procedimentos que, quando incorporados em um projeto, conseguem apoiar a execução e melhoria do processo, permitindo software de qualidade. (MALTA, 2023).

Ou seja, ambas as metodologias e práticas têm um objetivo principal em comum, que é responder de forma eficaz e rápida às mudanças, obtendo delas resultados positivos. Assim, as metodologias ágeis estão focadas na parte de desenvolvimento de software ou na gestão dos processos do projeto, dependendo da diferente metodologia aplicada. (MENDOZA; PIZZOLATO, 2023).

De acordo com Malta (2023) é possível até misturar a aplicação de diferentes metodologias e práticas para agilizar todo o processo, desde a gestão até a programação do código do software. Além do ponto de vista metodológico, existem certas práticas voltadas para a programação ágil, como aquelas associadas às áreas de design, desenvolvimento (DevOps) e testes (Testing), que detalham técnicas ágeis de desenvolvimento de software em um nível inferior.

Com a evolução das metodologias ágeis, surgiram uma infinidade de práticas que são incorporadas ao projeto dependendo da necessidade do momento, porém, o progresso da inclusão pode causar problemas e atrasos inesperados. Recomenda-se buscar qualidade no processo de programação que proporcione segurança e valor no produto final, onde essas práticas sejam um suporte para atingir esse objetivo de qualidade. (FELIPE, 2021).

Conforme definido por Malta (2023), as metodologias ágeis surgiram na década de 90, onde surgiram vários movimentos identificados com o nome de Metodologias Leves. Entre eles estão Extreme Programming (XP), Scrum, Software Craftsmanship, Lean Software Development, encontrados entre os mais populares.

Posteriormente, em fevereiro de 2001, um grupo de dezessete profissionais de desenvolvimento de software reconhecidos, e referências das metodologias leves existentes na época, reuniu-se em Utah (EUA), com o objetivo de determinar os valores e princípios que permitiriam às equipes desenvolver softwares e atender com mais precisão às necessidades do cliente e responder melhor às mudanças que possam surgir ao longo de um projeto de desenvolvimento. (MENDOZA; PIZZOLATO, 2023). De acordo com Souza:

> Os métodos ágeis passaram a ser utilizados após a realização de um encontro de dezessete desenvolvedores de software que estabeleceram princípios sobre como melhorar a entrega de software para os clientes. Com o encontro, um manifesto [Manifesto ágil 2001] foi confeccionado e alguns princípios foram estabelecidos. O manifesto possui as seguintes afirmações sobre as prioridades da realização do desenvolvimento de software:
>
> - Indivíduos e interações são mais importantes que processos e ferramentas.
> - Software em funcionamento é mais importante que documentação abrangente.
> - Colaboração com o cliente é mais importante que negociação de contratos.
> - Responder a mudanças é mais importante que seguir um plano.
>
> Essas sentenças são fundamentais para os princípios que abrangem o desenvolvimento ágil de software e com elas os desenvolvedores procuram entregar softwares de forma mais rápida, interativa e que consiga responder às mudanças propostas pelos clientes. (SOUZA, 2023, p 07).

O objetivo era oferecer uma alternativa aos processos tradicionais de desenvolvimento de software, caracterizados pela rigidez e dominados pela documentação. Neste encontro foi criada a Agile Alliance, uma organização sem fins lucrativos que tem como objetivo promover os valores e princípios da filosofia ágil e ajudar as organizações na sua adoção. (MALTA, 2023).

Também foi declarada a pedra angular do movimento ágil, conhecido como Manifesto Ágil. O movimento ágil na indústria de software veio à tona com o manifesto de desenvolvimento ágil de software publicado por um grupo de profissionais e consultores de

software em 2001, cujo objetivo foi dar a conhecer os valores que devem ser tidos em conta quando uma equipe de trabalho necessita desenvolver softwares de forma ágil, respondendo em tempo útil às diferentes alterações propostas pelo cliente durante o ciclo de vida do projeto.

Os valores em que este movimento se concentrou foram: aos indivíduos e à sua interação, acima dos processos e ferramentas; software que funciona, acima de documentação exaustiva; colaboração com o cliente, acima da negociação contratual; e a resposta à mudança, acima de seguir um plano. (MALTA, 2023).

Definições recentes de metodologias ágeis de desenvolvimento de software dizem que agilidade no desenvolvimento de software é a capacidade de uma equipe responder e incorporar mudanças nos requisitos do usuário de forma eficiente e eficaz durante o ciclo de vida do projeto. (MENDOZA; PIZZOLATO, 2023).

Também em definições mais específicas diz-se que são um conjunto de métodos evolutivos baseados na melhoria iterativa e em processos de desenvolvimento oportunos, onde cada iteração são miniprojetos independentes com atividades que incluem análise de requisitos, design, implementação e testes. (MALTA, 2023).

O manifesto Ágil como pode ser visto em Felipe (2021) é resumido em quatro valores e doze princípios. Entre os valores destacam-se a cooperação no trabalho em equipe, a eficiência e a eficácia para o desenvolvimento dos entregáveis entre outros.

Também não se deve esquece da metodologia Scrum. O termo Scrum é derivado do rugby, é um treinamento em que, com trabalho em equipe, se consegue a posse de bola. (SCHWABER; SUTHERLAND, 2015). Scrum foi desenvolvido para gerenciar o processo de desenvolvimento de sistemas introduzindo ideias de flexibilidade, adaptabilidade e produtividade. (MALTA, 2023).

Scrum não define nenhuma técnica específica de desenvolvimento de software para a fase de implementação. Ele se concentra em como os membros da equipe devem funcionar para obter flexibilidade nos sistemas ao lidar com um ambiente em constante mudança. (MENDOZA; PIZZOLATO, 2023).

A ideia principal do Scrum é que no desenvolvimento de sistemas geralmente existem variáveis que mudam durante o processo o que torna seu desenvolvimento imprevisível e complexo. (SCHWABER; SUTHERLAND, 2015). Tornando-se necessário

que haja flexibilidade no processo de desenvolvimento de forma que ele seja capaz de responder às mudanças. (MALTA, 2023).

O Scrum funciona da seguinte maneira: no início da iteração a equipe analisa o que precisa ser feito e, em seguida, seleciona o que acredita ser uma funcionalidade incremental e potencialmente entregável para o final da iteração. (SCHWABER; SUTHERLAND, 2015).

A equipe então faz o melhor que pode no restante da iteração para apresentar funcionalidades incrementais que as partes interessadas possam inspecionar. O coração do Scrum é a iteração, a equipe analisa os requisitos, leva em consideração a tecnologia disponível e avalia suas próprias habilidades e capacidades. (SCHWABER; SUTHERLAND, 2015).

Eles então determinam coletivamente como construir a funcionalidade, modificando sua abordagem diariamente à medida que encontram novas dificuldades e surpresas. (MALTA, 2023). A equipe percebe quais atividades devem ser realizadas e seleciona a melhor forma de realizá-las.

Este processo criativo é o coração da produtividade Scrum. Scrum tem três funções, uma delas é o dono do produto ou "Product Owner" que é responsável pelo sucesso do projeto do ponto de vista das partes interessadas, determina a visão do produto, gerencia as expectativas das partes interessadas, maximiza a lucratividade do produto, determinando prioridades de recursos à medida que o projeto avança. (MENDOZA; PIZZOLATO, 2023).

A equipe de desenvolvimento é responsável pela construção e qualidade do produto. A equipe é auto-organizada e tem como objetivo transformar as funcionalidades comprometidas em software funcional e de qualidade produtiva. (MALTA, 2023).

Por fim, o Scrum Master é responsável pelo processo Scrum e deve garantir que todos sigam suas regras e práticas. (SCHWABER; SUTHERLAND, 2015). Ele é responsável por ensinar Scrum a todos os envolvidos no projeto, bem como implementar o Scrum de forma que ele se encaixe na cultura de uma organização, entregando os resultados esperados. (FELIPE, 2021).

O Scrum possui três elementos principais, o Product Backlog, que é uma lista de itens ou características do produto a serem construídos, mantidos e priorizados pelo Product Owner, um Sprint Backlog, que é o conjunto de itens do Product Backlog que serão

construídos na iteração ou Sprint, e por fim o incremento funcional potencialmente entregável, que é a entrega do Sprint. (MALTA, 2023).

As principais características do Scrum são que o desenvolvimento de software é realizado em iterações que não duram mais de 30 dias denominadas Sprint, e o resultado de cada Sprint é uma entrega que agrega funcionalidade e é mostrada ao cliente. (MENDOZA; PIZZOLATO, 2023).

A segunda característica é que durante as fases do projeto são realizadas reuniões, destacando-se a reunião realizada diariamente pela equipe de desenvolvimento para coordenação e integração. Conforme definido pelo autor Malta (2023) Scrum é um framework que nos permite encontrar práticas emergentes em domínios complexos, como a gestão de projetos de inovação.

Não é um processo completo, muito menos uma metodologia. Em vez de fornecer uma descrição completa e detalhada de como devem ser realizadas as tarefas de um projeto, gera um contexto relacional e iterativo de constante inspeção e adaptação para que os envolvidos criem seu próprio processo. (SOUZA, 2015).

Isso ocorre porque não existem melhores ou boas práticas em um contexto complexo. A equipe de desenvolvimento é apoiada por duas funções: o Scrum Master e o Product Owner. (SCHWABER; SUTHERLAND, 2015). O Scrum Master é quem garante a utilização do Scrum, a remoção de impedimentos e auxilia a equipe a atingir seu maior nível de desempenho possível. (MALTA, 2023).

É possível ter considerado um coach ou facilitador encarregado de acompanhar a equipe de desenvolvimento. O Product Owner é quem representa o negócio, as partes interessadas, o cliente e os usuários finais. Há a responsabilidade de liderar a equipe de desenvolvimento em direção ao produto certo. (MENDOZA; PIZZOLATO, 2023).

O andamento dos projetos utilizando Scrum é feito e verificado em uma série de iterações chamadas Sprints. (SCHWABER; SUTHERLAND, 2015). Esses Sprints têm duração fixa e pré-estabelecida de no máximo um mês. No início de cada Sprint a equipe de desenvolvimento realiza um compromisso de entregar uma série de funcionalidades ou características do produto em questão. (MALTA, 2023).

No final do Sprint, espera-se que estas funcionalidades comprometidas estejam concluídas, o que implica a sua análise, design, desenvolvimento, testes e integração no produto. É quando é realizada uma reunião de revisão do produto construído durante o

Sprint, onde a equipe de desenvolvimento mostra o que foi construído para o Product Owner e qualquer parte interessada em participar. (FELIPE, 2021).

O feedback obtido nesta reunião poderá ser incluído entre as funcionalidades a serem construídas em Sprints futuros. Por sua vez, conforme mencionado por Malta (2023), OpenUP é um processo ágil e unificado, que contém o conjunto mínimo de práticas que ajudam as equipes a serem mais eficazes no desenvolvimento de software.

OpenUP adota uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento de software. É um processo iterativo mínimo, completo e extensível que pode ser usado como está ou expandido para atender a uma ampla variedade de tipos de projetos.

Caracteriza-se por ser iterativo e incremental, focado na arquitetura e orientado por casos de uso. (MENDOZA; PIZZOLATO, 2023). Está organizado em quatro áreas de conteúdo principais: comunicação e colaboração, intenção, solução e administração.

OpenUp está organizado em duas dimensões diferentes, mas inter-relacionadas: método e processo. O conteúdo do método é onde os elementos do método (funções, tarefas, artefatos e diretrizes) são definidos, independentemente de como são utilizados no ciclo de vida do projeto. (MALTA, 2023).

O processo é onde os elementos do método são aplicados de maneira ordenada ao longo do tempo. Muitos ciclos de vida para diferentes projetos podem ser criados a partir do mesmo conjunto de elementos de método. (FELIPE, 2021). O OpenUp é caracterizado por quatro princípios fundamentais que se apoiam mutuamente: colaboração para alinhar interesses e compreensão partilhada; equilíbrio para confrontar prioridades para maximizar valor para os stakeholders; foco na articulação da arquitetura para facilitar a colaboração técnica, reduzir riscos e minimizar excessos e trabalho extra; e evolução contínua para reduzir riscos, demonstrar e obter resultados.

O desenvolvimento de um produto de software é visto como um processo, que requer a aplicação de técnicas e procedimentos que garantam a qualidade do produto. Esses conjuntos de etapas são conhecidos como métodos de desenvolvimento de software, que são recursos sistemáticos para orientar os desenvolvedores na construção de software confiável e de qualidade, de forma produtiva, disciplinada e previsível. (MENDOZA; PIZZOLATO, 2023).

Os métodos permitem representar as etapas do ciclo de vida (requisitos, análise, design, codificação, testes, implementação e manutenção), bem como ordenar atividades e designar funções. Ao mesmo tempo, a principal intenção da engenharia de software é melhorar a qualidade dos seus produtos para garantir que sejam competitivos e, assim, ajustar-se aos requisitos estabelecidos pelo cliente (usuários finais). (SOUZA, 2015).

Por sua vez, Malta (2023) define a qualidade do produto de software como o grau em que esse produto satisfaz os requisitos dos seus utilizadores, agregando assim valor. Da mesma forma, o modelo de qualidade do produto é composto por oito características de qualidade: adequação funcional, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, manutenibilidade, portabilidade e segurança.

Geralmente, a engenharia de software enfatiza o controle de processos através de uma definição rigorosa de papéis, atividades e artefatos, este esquema tradicional tem se mostrado eficaz e necessário em grandes projetos onde geralmente é necessário um alto grau de formalismo em seu desenvolvimento, porém, a realidade dos projetos está localizada em um ambiente onde é necessário reduzir drasticamente os tempos de desenvolvimento mantendo a alta qualidade. (MENDOZA; PIZZOLATO, 2023).

Considera-se um projeto de sucesso aquele que é concluído no prazo e dentro do orçamento, com todas as características e funções especificadas no início. Consequentemente, em relação ao desenvolvimento de projetos de software, existe uma variedade de propostas metodológicas que promovem e oferecem a utilização de ferramentas e artefatos que são levados em consideração à medida que a aplicação vai sendo desenvolvida para garantir o sucesso do projeto. (MALTA, 2023).

Estas visam reduzir custos, aproveitar os recursos disponíveis, medir o progresso e melhorar a qualidade do sistema. As metodologias de desenvolvimento de software podem ser classificadas em dois grupos, as baseadas em planos ou tradicionais, que enfatizam a documentação e o controle do projeto, e as ágeis voltadas para pequenas equipes de desenvolvimento, que focam no fator humano e consideram o cliente como elemento principal do projeto. (FELIPE, 2021).

Desenvolvimento softwares seguros

A criticidade e tendência de utilização dos atuais sistemas de informação nos diferentes domínios da sociedade determina que, embora seja importante garantir que o

software é desenvolvido de acordo com as necessidades do utilizador, é também importante garantir que o mesmo é seguro. (SOUZA, 2023).

Pelo exposto, é relevante compreender o conceito de software seguro, para isso Mendoza e Pizzolato (2023) propõe-se o triângulo de segurança, onde cada vértice indica um aspecto fundamental do serviço a ser implementado: funcionalidade, segurança e experiência de usuário.

Este triângulo é polémico para os utilizadores e, ao mesmo tempo, é causa de grande esforço, tanto em recursos como em tempo, de todos os técnicos relacionados com a implementação de um novo serviço. Embora o ideal seja que este triângulo seja equilátero, a realidade mostra que cada vez que um dos vértices é enfatizado, ele se afasta, produzindo uma forte diminuição nos outros dois. (SOUZA, 2023).

Para melhor compreender a importância e o papel da segurança no triângulo, os seguintes aspectos são apontados como suas propriedades fundamentais: confidencialidade, que refere-se ao fato de que o software deve garantir que qualquer uma de suas características, incluindo as suas relações com o ambiente de execução e os seus utilizadores, os ativos que gere e/ou o seu conteúdo são acessíveis apenas a entidades autorizadas e inacessíveis a outras; integridade, em que o software e os ativos que eles gerenciam são resilientes e flexíveis à subversão; disponibilidade em que o software deve estar operacional e acessível aos seus usuários autorizados sempre que necessário; rastreabilidade, todas as ações relevantes relacionadas à segurança de uma entidade que atua como usuário devem ser registradas e rastreadas para estabelecer responsabilidades; e não-repúdio que é a capacidade de impedir que uma entidade agindo como usuário negue ou negue responsabilidade por ações que foram executadas.

As consequências da violação da segurança de software podem ser descritas em termos dos efeitos sobre estas propriedades fundamentais, razão pela qual a gestão da segurança no desenvolvimento de software é um fator cada vez mais determinante na competitividade das organizações no ambiente global. (SOUZA, 2023).

Uma vez que a segurança informática, um termo geral que abrange uma grande área da computação e do processamento de informação, é frequentemente dividida em três categorias fundamentais denominadas controles, com as quais define os principais objetivos de uma implementação de segurança adequada. (MENDOZA; PIZZOLATO, 2023).

A segurança informática e a implementação dos controles que ela impõe tornaram-se uma despesa quantificável e justificável para todos os orçamentos de tecnologia porque as indústrias dependem de sistemas e redes informatizadas para executar suas operações e transações comerciais diárias. (SOUZA, 2023).

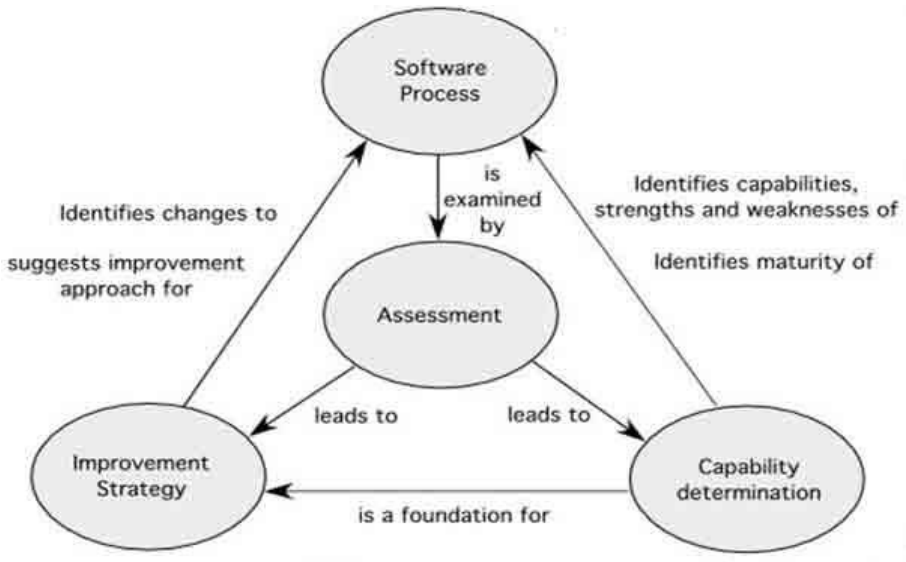
É por isso que consideram seus dados como uma parte importante de seus ativos globais. Na verdade, muitos termos e medidas entraram no vocabulário empresarial diário, como custo total de propriedade e qualidade dos serviços. (FELIPE, 2021).

Usando essas medidas, as indústrias calculam aspectos como integridade de dados e alta disponibilidade como parte do planejamento de processos e custos de gerenciamento. (BEZERRA, 2019). Em alguns setores, como o comércio eletrônico, a disponibilidade e a confiabilidade dos dados podem fazer a diferença entre o sucesso e o fracasso.

Infelizmente, proteger sistemas e redes pode ser uma proposta difícil, exigindo um conhecimento complexo de como uma organização confia, utiliza, manipula e transmite as suas informações. (SOUZA, 2023). Compreender a forma como a organização gere o seu negócio e as pessoas que a compõem são essenciais para implementar um plano de segurança adequado.

Consequentemente, a gestão de riscos e a garantia da informação levaram a trabalhar na evolução da melhoria dos processos de software, também conhecida pela sigla SPI (Software Process Improvement), onde foram definidos modelos padronizados de boas práticas. (MENDOZA; PIZZOLATO, 2023).

Imagem 01 – Estrutura do SPI



Fonte: googleimagens.com

A garantia de software, em palavras simples, é a confiança de que um sistema atende a todos os seus requisitos de segurança. Principalmente aqueles de interesse para usuários e proprietários de produtos, esta confiança é baseada em evidências específicas coletadas e avaliadas por meio de técnicas de garantia. (SOUZA, 2023).

Software seguro de acordo com Felipe (2021) referenciado é um software que continua a funcionar corretamente mesmo sob ataques maliciosos intencionais. Mendoza e Pizzolato (2023) define software seguro como software que não pode ser alterado intencionalmente ou forçado a falhar.

Em suma, software que permanece confiável (adequado e previsível), apesar dos esforços intencionais para comprometer a sua confiabilidade. Além disso, Souza (2023) nos mostra diversas definições de garantia de software, entre as quais se destaca a do Departamento de Defesa dos Estados Unidos, que diz que garantia de software é o nível de confiança em que as funções do software são conforme o esperado e que estão livres de interferências intencionalmente ou vulnerabilidades projetadas involuntariamente e livres de vulnerabilidades inseridas como parte do software.

O desenvolvimento seguro de softwares é um aspecto crucial que deve ser considerado durante todo o ciclo de vida de um projeto. Especialmente em equipes ágeis, onde a eficiência e a agilidade são fundamentais, é essencial garantir que a segurança não seja negligenciada.

Souza (2023) também afirmar que é preciso priorizar o sistema de segurança, uma vez que o volume de ciberataques está crescendo rapidamente. O desenvolvimento seguro atua como uma camada de proteção extra, minimizando riscos de invasão, vazamento de dados e outras ameaças.

Os benefícios incluem atendimento à legislação de proteção de dados, redução de retrabalho e aumento da confiança do Mendoza e Pizzolato (2023) entendem que o desenvolvimento seguro não precisa ser disruptivo. Pelo contrário, quando incorporado desde o início, ele fortalece a equipe ágil, protege o negócio e constrói confiança no mercado.

A correção por construção (CbyC), segundo Souza (2023), é um método eficaz para desenvolver softwares que exige um nível crítico de segurança e que também é demonstrável. Os principais objetivos desta metodologia são obter uma taxa mínima de defeitos e alta resiliência à mudança; que são alcançados devido a dois princípios

fundamentais: dificultar muito a introdução de erros e garantir que os erros sejam removidos assim que forem injetados.

A CbyC busca produzir um produto isso desde o início está correto, com requisitos de segurança rigorosos, com uma definição muito detalhada do comportamento do sistema e um design sólido e verificável. (MENDOZA; PIZZOLATO, 2023).

Quanto às fases da metodologia CbyC, estas combinam métodos formais com desenvolvimento ágil; utiliza notações precisas e desenvolvimento incremental que permite mostrar o progresso para receber feedback e avaliação do produto. (FELIPE, 2021).

Segundo Souza (2023) o desenvolvimento de segurança (SDL), é um processo para melhorar a segurança do software proposto pela empresa Microsoft em 2004; com dezesseis atividades focadas em melhorar a segurança do desenvolvimento de um produto de software.

As práticas propostas pelo SDL vão desde uma etapa de treinamento em questões de segurança, passando por análise estática, análise dinâmica, fuzz test do código até ter um plano de resposta a incidentes. Um dos principais recursos do SDL é a modelagem de ameaças, que ajuda os desenvolvedores a encontrar partes do código onde é provável que existam vulnerabilidades ou ataques. (MENDOZA; PIZZOLATO, 2023).

Existem duas versões do SDL, a versão rígida e a voltada para o desenvolvimento ágil. As diferenças são que o segundo desenvolve o produto de forma incremental e na frequência de execução das atividades para garantir a segurança. (SOUZA, 2023).

A versão rígida do SDL é mais apropriada para equipes e projetos de desenvolvimento maiores e não são suscetíveis a alterações durante o processo. Agile SDL é recomendado para desenvolvimento de aplicativos web ou baseados na web. (BEZERRA, 2019).

No gráfico 10 encontra-se o fluxo das fases da metodologia SDL, vale destacar a existência de uma etapa anterior aos requisitos, focada no treinamento de segurança e a última etapa do processo que é responsável por monitorar o produto caso de um incidente de segurança. (SOUZA, 2023).

CONSIDERAÇÕES FINAIS

A presente investigação foi realizada com o intuito de dar um panorama de como estão sendo realizados os projetos de software quando se leva em consideração o desenvolvimento de software seguro utilizando metodologias ágeis. Adicionalmente, a

metodologia ágil mais utilizada é o Scrum devido à adaptação que lhe pode ser dada, apesar disso, estudos e análises consideram que a metodologia não é robusta o suficiente para lidar com requisitos ou histórias de usuários focadas em segurança de software, mas a observam como uma metodologia para realizar projetos de curto ou médio prazo.

Para as metodologias Open Up e XP existe uma relação semelhante já que a experiência na prática não é significativa e embora as considerem como eficientes e metodologias competentes, observa-se que não optam por adaptá-las ao desenvolvimento de software seguro.

As implicações de segurança no desenvolvimento de projetos com metodologias ágeis são as seguintes: os artefatos de segurança gerados concentram-se em maior medida em análise de risco e testes baseados em risco; ao utilizar metodologias ágeis, não se reconhece claramente em qual das etapas, dependendo da metodologia utilizada, os temas relacionados à segurança de software devem começar a ser incluídos.

Em termos gerais, os resultados obtidos mostram as desvantagens que as empresas apresentam em termos de desenvolvimento seguro de software, formação de pessoal especializado no assunto, estimativa e definição para a execução de projetos que contenham uma elevada componente de segurança, falta de conhecimento de ferramentas para verificação de falhas relacionadas à segurança no sistema e desenho de testes focados na detecção de erros de segurança.

Tendo em conta o trabalho realizado, sugerem-se incluir outros aspectos relacionados ao desenvolvimento de software seguro, que não foram contemplados neste trabalho. Da mesma forma, é necessário considerar também, a segurança de acordo com as linguagens de programação utilizadas.

REFERÊNCIAS BIBLIOGRÁFICAS

BEZERRA, Carlos Magnum Matias. **Políticas para desenvolvimento de software seguro em times ágeis**. 2019.

FELIPE, Lais Pereira. **O Estado da Arte em Metodologias e Práticas ágeis de desenvolvimento de Software.** PUC/ Goiás. Goiânia, 2021. Disponível em: https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/3639/1/LaisFelipe_Praticas_Ageis_de_Desenvolvimento.pdf. Acesso em: 25/02/2024.

LAKATOS, E, M. MARCONI, M. A. **Metodologia cientifica.** 7º Edição, São Paulo: Atlas, 2017.

MALTA, Matheus Baptista. **Melhoria de indicadores ambientais com a utilização de metodologias ágeis.** 2023. 54 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Recursos Hídricos e do Meio Ambiente) - Escola de Engenharia, Universidade Federal Fluminense, Niterói, 2023. Disponível em: http://app.uff.br/riuff/handle/1/31159. Acesso em: 24/02/2024.

MENDOZA, Daniela Milagros Quenaya; PIZZOLATO, Nélio Domingues. **Gerenciamento e planejamento de projetos de software usando metodologias ágeis:** um estudo de caso. Revista De Gestão E Secretariado, 14(11), 20568–20585. 2023. Disponível em: https://doi.org/10.7769/gesec.v14i11.2952. Acesso em: 21/02/2024.

SCHWABER, Ken; SUTHERLAND, Jeff. Guia do Scrum. **Um guia definitivo para o Scrum:** As regras do jogo. Jul. 2013. Disponível em: Acesso em dezembro, 2015.

SOUZA, Luciano Malaquias de. **Método Ágil XP** (Extreme Programming). Academos, Revista Eletrônica da FIA, Vol.III. Jul - dez. 2007. Disponível em: Acesso em outubro, 2015.

SOUZA, Rafael Carneiro Reis de. **Adotando modelagem de ameaças em projetos ágeis de desenvolvimento de software** / Rafael Carneiro Reis de Souza. - Recife, 2023. 21 p.: il., tab. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/49925/1/TCC%20Rafael%20Carneiro%20Reis%20de%20Souza.pdf. Acesso em: 26/02/2024.