Brazilian Journal of Development

Técnicas e ferramentas para detecção de vulnerabilidades em ambientes de desenvolvimento ágil de software

Techniques and tools for detection of vulnerabilities in software agile development environments

DOI:10.34117/bjdv6n6-081

Recebimento dos originais:08/05/2020
Aceitação para publicação:04/06/2020

Ligia Cássia M. C. Santos
Mestre em Sistemas de Informação pela Universidade de São Paulo
Endereço: Rua Arlindo Bettio, 1000 - CEP 03828.000 - Vila Guaraciaba - São Paulo/SP
Email: ligia.santos01@usp.br

Edmir Parada Vasques Prado
Doutor em Administração pela Universidade de São Paulo
Endereço: Rua Arlindo Bettio, 1000 - CEP 03828.000 - Vila Guaraciaba - São Paulo/SP
Email: eprado@usp.br

Marcos Lordello Chaim
Doutor em Engenharia Elétrica pela Universidade Estadual de Campinas
Endereço: Rua Arlindo Bettio, 1000 - CEP 03828.000 - Vila Guaraciaba - São Paulo/SP
Email: chaim@usp.br

RESUMO
Os sistemas de informação precisam ser seguros para atingir seus objetivos. Portanto, durante o desenvolvimento, é necessário detectar possíveis vulnerabilidades presentes no software. Este artigo tem como objetivo descrever o uso de técnicas e ferramentas de detecção de vulnerabilidades (TFDV) em ambientes complexos e sua relação com a qualidade de software e modelos de serviço em equipes que usam métodos ágeis. Para isso, foi realizada uma pesquisa listando 18 técnicas e ferramentas que usavam três processos bem conhecidos de desenvolvimento seguro de software. A pesquisa foi aplicada a 76 membros de equipes de desenvolvimento ágil de software, que já implantaram, estão em processo de implantação, ou vão implantar TFDV nos projetos. A partir dos dados coletados, foi possível descrever o uso de TFDV em ambientes complexos de desenvolvimento e identificar associações entre a adoção da qualidade de software e o uso de TFDV. Para tanto, foi utilizado o Teste Exato de Fisher para analisar e interpretar as associações encontradas

Palavras-Chaves: Desenvolvimento de Software, Detecção de Vulnerabilidades, Métodos Ágeis

Brazilian Journal of Development

ABSTRACT

Information systems need to be safe to achieve their goals. Thus,during development one needs to detect possible vulnerabilities present in the software. This paper aims to describe the use of vulnerability detection techniques and tools (VDTT) in complex environments and its relationship to software quality and service models in teams that use agile methods. To do so, a survey was carried out listing 18 techniques and tools that used three well known processes of secure software development. The survey was applied to 76 members of agile software development teams who have already deployed, are in the process of deploying, or are going to deploy vulnerability detection techniques and tools in the projects. From the data collected, we were able describe the use of VDTT in complex development environments and associations between the adoption of software quality and service models and the use of VDTTs. For this purpose, Fisher's exact test was used to analyze and interpret the associations found.

keywords: Software Development, Vulnerability Detection, Agile Methods.

1 INTRODUÇÃO

Atualmente, as organizações operam em ambientes complexos, com mudanças constantes e forte competitividade. Por essa razão, elas precisam responder de forma ágil às pressões do ambiente.

Por outro lado, os processos de desenvolvimento de software que planejam especificar completamente os requisitos antecipadamente, ou seja, orientados a planos, são inadequados para ambientes que sofrem constantes mudanças (SOMMERVILLE, 2011).

Para ambientes com constantes mudanças uma abordagem ágil é recomendável. Esta abordagem, de acordo com Beck (2000), possui valores e princípios que devem ser seguidos. Entretanto, devido à agilidade do processo de desenvolvimento e à pressão na entrega, alguns softwares não têm uma análise completa da segurança para cada fase do ciclo de vida e com isso podem surgir vulnerabilidades (OWASP, 2017). A maioria das vulnerabilidades nos sistemas de software são causadas pela falta de aplicação de políticas de segurança no projeto (OWASP, 2017).

Para evitar a ocorrência de vulnerabilidades, segundo Howard e Lipner (2006), é necessário analisar: o uso de práticas de segurança pelas equipes; as habilidades dos membros; e as necessidades de formação da equipe. Howard e Lipner (2006) apontam que práticas de segurança incorporadas ao ciclo de vida de desenvolvimento de software podem diminuir a probabilidade de existirem vulnerabilidades no software.

Pelo fato de os métodos ágeis trabalharem com incrementos rápidos, muitas vezes a segurança não é verificada adequadamente. Além disso, esses métodos têm se tornado

prevalentes na indústria de software por utilizar práticas baseadas com pouca documentação, interação com o cliente, respostas rápidas a mudanças, muita comunicação verbal e foco na geração de código executável da aplicação (BECK, 2000).

Desta forma, é importante avaliar as práticas de segurança aplicadas pelos membros de equipes que usam métodos ágeis porque os ataques cada vez mais frequentes em softwares exigem das organizações políticas de segurança efetivas (VIANNA; FERNANDES, 2015).

A literatura destaca diversos processos de desenvolvimento de software seguro tais como Processo de McGraw (2006), Howard e Lipner (2006) e OWASP (2016). Entretanto, nenhum deles está voltado exclusivamente para métodos ágeis. Dentro deste contexto, o objetivo deste trabalho é descrever a relação entre técnicas e. ferramentas de detecção de vulnerabilidades TFDV no contexto de desenvolvimento ágil de software e a sua associação com o uso de modelos de qualidade de software e serviços em equipes que utilizam métodos ágeis.

Para atingir este objetivo os seguintes objetivos específicos foram definidos: (1) descrever as TFDV existentes na literatura; (2) listar as TFDVs; (3) identificar a associação entre o uso de métodos ágeis e o uso de TFDV; e (4) identificar a associação entre o uso de modelos de qualidade de software e serviços e o uso de TFDV.

A estrutura do artigo é descrita a seguir. A Seção 2 descreve os trabalhos relacionados. A Seção 3 apresenta a fundamentação teórica. Em seguida, é apresentado o modelo de pesquisa na Seção 4 e o método de pesquisa empregado na Seção 5, formando a base para a apresentação da análise e discussão dos resultados na Seção 6. A Seção 7 encerra o artigo com o apontamento das considerações finais.

2 TRABALHOS RELACIONADOS

A partir da revisão bibliográfica realizada, foram encontrados trabalhos que procuram identificar e avaliar o uso de técnicas de segurança em empresas que utilizam métodos ágeis por meio de *surveys*, entrevistas e estudos de casos. Outra abordagem relacionada com essa pesquisa mostra trabalhos que propõem extensões dos métodos ágeis para inclusão de atividades de segurança, bem como a comparação entre atividades de segurança dos processos tradicionais e dos métodos ágeis.

A seguir são descritos os trabalhos obtidos, de acordo com essas duas abordagens.

Brazilian Journal of Development

2.1 AVALIAÇÕES EMPÍRICAS

Vários trabalhos conduziram avaliações empíricas do estado da prática do uso de TFDVs com profissionais ou em ambientes de desenvolvimento. Márquez *et al*. (2015) verificam se os artefatos de segurança em desenvolvimento de software são relevantes no contexto do projeto e como eles são usados. Bartsch (2011) descreve *insights* sobre os profissionais especialistas em segurança em desenvolvimento ágil a partir de pesquisas exploratórias, qualitativas e resultados de entrevistas.

Baca *et al.* (2015) apresentam um estudo realizado com desenvolvedores da Ericsson, empresa global do segmento de telecomunicações, no qual foram avaliadas diferentes atividades de segurança aplicadas nas fases de desenvolvimento dentro de um projeto usando métodos ágeis. As atividades de segurança são oriundas de três processos de desenvolvimento de software seguros: *Cigatel Touchpoints, Common Criteria* e *Microsoft SDL*. A melhoria da segurança no contexto ágil é o tema proposto nos trabalhos de Cruzes *et al*. (2017) e OYETOYAN *et al*. (2016). Neste estudo, os autores utilizaram um instrumento de pesquisa para realizar a medição de habilidades de segurança de software, uso e necessidades de treinamento em equipes ágeis.

2.2 EXTENSÕES DE MÉTODOS ÁGEIS E COMPARAÇÕES ENTRE PROCESSOS

Outros trabalhos propõem extensões nos métodos ágeis para incorporar atividades de segurança. Por exemplo, Othmane *et al*. (2014) propõem um método para a garantia de segurança de incrementos de software. O artigo aborda a possibilidade de estender o processo de desenvolvimento ágil para produzir software aceitavelmente seguro em cada iteração. Os processos CLASP e SDL são avaliados e comparados por Gregoire *et al*. (2007). Assim, é realizada uma comparação das atividades dos processos, considerando cada fase do ciclo de vida de desenvolvimento de software.

O estudo de Baca *et al*. (2015) também incorpora atividades de segurança em processo tradicionais de desenvolvimento de software seguro. Oyetoyan *et al*. (2016) sugerem a inclusão de técnicas e ferramentas de segurança nas fases de desenvolvimento, bem como a de um especialista em segurança para orientar os desenvolvedores.

2.3 COMPARAÇÃO COM OS TRABALHOS SELECIONADOS

Esta pesquisa se diferencia das demais listadas da literatura quanto ao seu objetivo que é descrever a relação entre técnicas e ferramentas de detecção de vulnerabilidades TFDV em

ambientes de desenvolvimento ágil de software e a sua associação com o uso de modelos de qualidade de software e serviços. Foram encontrados diversos trabalhos na literatura que abordam a incorporação de segurança em métodos ágeis, porém, nenhum deles descrevem a associação entre a adoção de modelos de qualidade de software e serviços em ambientes de desenvolvimento ágil de software, e o uso de técnicas e ferramentas de detecção de vulnerabilidades.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 VULNERABILIDADES

Uma vulnerabilidade em software é um conjunto de condições que podem levar à violação de uma política de segurança (KHAIM *et al*., 2016). Tais condições podem ser oriundas da má especificação de requisitos de segurança, problemas de design, práticas de codificação insegura, falhas nas atividades de garantia de segurança ou problemas de manutenção do sistema (HOWARD; LIPNER, 2006).

A vulnerabilidade é uma fraqueza na aplicação que permite que um atacante cause danos aos *stakeholders* de um sistema (OWASP, 2017). Em um sistema, procedimentos de segurança e controles internos ou de implementação que podem ser explorados por ameaças são considerados vulnerabilidades (SEACORD; HOUSEHOLDER, 2005).

É, portanto, importante detectar vulnerabilidades durante o desenvolvimento de software, pois informações perdidas, utilizadas incorretamente e acessadas por pessoas não autorizadas podem prejudicar uma organização. De acordo com Bishop (2003) a detecção de vulnerabilidades ocorre ao verificar a eficácia das medidas preventivas em relação a potenciais ataques.

Quando é mencionado o termo vulnerabilidade em software, vários outros correlatos vêm juntamente a este, como é o caso de ameaças (NIST, 2009). Uma ameaça é uma ocorrência que pode acarretar dano para o sistema ou organização em um ativo ou grupos de ativos (GTIESKAMP *et al*., 2002). É uma possível violação de um sistema computacional que pode ser acidental ou intencional. Uma ameaça acidental pode ser, por exemplo, uma possível falha no hardware ou software. Já a ameaça intencional está associada à premeditação, como um monitoramento não autorizado por hackers (STALLINGS, 2012). Assim, uma ameaça está relacionada ao termo vulnerabilidade, pois esta é definida como uma condição que, quando explorada por um atacante, pode resultar em uma violação de segurança (PINHEIRO, 2011).

Um ataque ocorre quando uma ameaça intencional é realizada por motivos variados como: ganhos financeiros, venda de informações, espionagem ou sabotagem (STALLINGS, 2012). Uma das possíveis causas de ataque é a dificuldade em desenvolver softwares com a devida segurança (CERT.br, 2012). Outro termo associado a vulnerabilidades é o risco, o qual pode ser definido como a probabilidade da ocorrência de uma ameaça (VIANNA; FERNANDES, 2015).

Uma vulnerabilidade também está ligada ao termo falha que pode ser definido como a incapacidade de executar uma função requerida dentro dos limites especificados (VERDON; MCGRAW, 2004). As falhas geralmente são classificadas em físicas, aquelas de que padecem os componentes, e humanas, as que compreendem a falhas de projeto e de interação.

As principais causas de falhas são problemas de especificação, problemas de implementação, componentes defeituosos, além de variações ambientais, como temperatura, pressão, umidade e problemas operacionais (GROUP, 2010).

É possível associar uma vulnerabilidade ao termo malicioso, que pode ser um código adicionado, alterado ou removido de um sistema de software, a fim de causar intencionalmente danos ou subverter a função pretendida do sistema (WEBER, 2003).

Uma vulnerabilidade também está ligada ao termo engano que é definido como uma ação humana que produz um defeito (VERDON; MCGRAW, 2004). Um defeito pode ser introduzido em qualquer uma das fases do ciclo de desenvolvimento de software, como na especificação de requisitos, design, implementação, verificação ou manutenção (MCGRAW; MORRISSETT, 2000).

3.1.1 Principais Vulnerabilidades de Software

Segundo OWASP (2017), aplicações seguras são resultado da decisão de produzi-las por parte da organização. A OWASP listou as 10 principais vulnerabilidades em aplicações *web*:

• **Injeções**. Injeções de falhas, particularmente injeções SQL (*Structured Query Language*), são comuns em aplicações *web*. Elas acontecem quando os dados que o usuário dá como entrada são enviados como parte de um comando ou consulta.

• **Falha de autenticação e gerenciamento de sessão**. Funções de aplicação como autenticação e gerenciamento de sessão, muitas vezes, não são implementadas corretamente, permitindo que possíveis ataques comprometam senhas, chaves ou *tokens* de sessão.

• **Exposição de dados sensíveis**. Muitas aplicações web não protegem adequadamente seus dados sensíveis, tais como informações de cartões de crédito, documentos e credenciais de autenticação.

• **Entidades XML**. São utilizados para compartilhar arquivos, realizar varredura de portas internas e ataques de negação de serviço.

• **Falta de controle de nível de acesso**. As aplicações web, que verificam credenciais dos seus usuários precisa executar verificações de controle de acesso no servidor.

• **Ausência de configurações**. É necessário definir e implantar uma configuração segura em aplicações, *frameworks*, servidor de aplicativos, servidor web, servidor de banco de dados, entre outros

• ***Cross site scripting – XSS***. Ocorre em aplicações que passam informações do usuário ao navegador sem validar ou codificar seu conteúdo.

• **Reversão insegura**. Executa remotamente trechos de código. São usados para escalonar e alterar privilégios.

• **Componentes com vulnerabilidades conhecidas**. Uso de bibliotecas, componentes e *frameworks* que possuem vulnerabilidades pode permitir ataques.

• **Ausência de *logs* para monitoramento**. A ausência de *log* para monitoramento dos acessos e detecção de possíveis violações na aplicação.

3.1.2 Principais Vulnerabilidades de Software

De acordo com Bishop (2003), ao analisar aspectos de segurança de software é importante fazer uma distinção entre política e mecanismo. Uma política de segurança é uma declaração do que é ou não é permitido pela aplicação. Já um mecanismo de segurança é um método, ferramenta ou procedimento para pôr em prática uma política de segurança.

Desta forma, para atingir o objetivo do estudo, primeiramente, por meio de uma revisão bibliográfica, foram identificadas e descritas 18 técnicas e ferramentas de segurança que são comuns aos seguintes processos: OWASP (2016), Processo de McGraw (2006) e atividades de Howard e Lipner (2006). Elas foram classificadas em função das fases do ciclo de vida de desenvolvimento de software: análise de requisitos (A.R.) e modelagem, design, codificação, testes e produção, como mostra a Tabela 1.

Brazilian Journal of Development

Tabela 1. TFDV classificadas por fase do ciclo de vida

Análise de requisitos e modelagem	**Codificação**
• Especialista em segurança Codificação segura	• Codificação segura
• Abuse / misuse; cases / stories	• Análise estática de código
• Requisitos de segurança	• Revisão de segurança de código
• Modelagem de ameaças	• Ferramenta de revisão de código
• Modelagem de riscos	• Testes
• Ferramenta de modelagem de riscos e ameaças	• Análise dinâmica de código
• *Design*	• Teste de penetração
• *Design* de segurança	• *Fuzz testing*
• Contramedidas de segurança	• Teste baseado em riscos
• Avaliação de vulnerabilidades	• Produção
• - X -	• Gestão de respostas a incidentes

Fonte: o próprio autor

3.2 MODELOS DE QUALIDADE DE SOFTWARE E SERVIÇOS

Entre as modelos de referência existentes na literatura, Cristofoli *et al*. (2012) destacaram o *Information Technology Infrastructure Library* (ITIL) como um dos mais utilizados no gerenciamento da tecnologia da informação (TI). Especificamente na área de qualidade de software e serviços, os modelos mais conhecidos e citados são o *Capability Maturity Model Integration* (CMMI) e o MPS.br, que serão apresentados a seguir.

3.2.1 Information Technology Infrastructure Library

O ITIL é uma biblioteca de infraestrutura de TI que reúne as melhores práticas para a gestão dos serviços de TI (AXELOS, 2018). Ele mapeia o ciclo de vida de serviços de TI por meio de cinco pilares.

• **Estratégia do serviço**. Trata das decisões estratégicas relacionadas aos serviços que serão desenvolvidos para o alcance dos objetivos de negócio.

• **Desenho de serviço**. Organiza as decisões estratégicas, elaborando as descrições das especificações dos serviços.

• **Transição de serviço**. Gerencia as implantações de serviços e transfere os serviços definidos para o ambiente de produção.

• **Operação do serviço**. São processos usados diariamente e que mantêm os serviços funcionando.

• **Melhoria contínua do serviço**. Promove a melhoria dos serviços, aplicando o ciclo PDCA (Plan-Do-Check-Act).

Esses cinco pilares ainda são compostos por 26 processos e quatro funções que contribuem para a aplicação dos serviços de acordo com as áreas, fases do ciclo de vida e funções. Os principais benefícios proporcionados pelo uso do ITIL são: eficiência operacional, redução dos custos e esforços desprendidos pela área de TI; cumprimento das atividades; e alinhamento do setor de TI com a área de negócios.

3.2.2 Capability Maturity Model Integration

O CMMI (*Capability Maturity Model Integration*) é uma abordagem para a melhoria de processos. Ele fornece às organizações elementos essenciais para promover processos eficazes e pode ser usado para guiar a melhoria em um projeto, de uma divisão ou em uma organização inteira (CMMI, 2018). O modelo é composto por quatro áreas de conhecimento, a saber: Engenharia de Sistemas (CMMI-SE), Engenharia de Software (CMMI-SW), Seleção de Fornecedores (CMMI-SS) e Desenvolvimento Integrado de Produto e Processo (CMMI-IPPD). Assim, o CMMI estabelece cinco níveis de maturidade para desenvolvimento de produtos, prestação de serviços e aquisição:

- **Nível 1 - Inicial**. Processos são imprevisíveis, pouco controlados e reativos.
- **Nível 2 - Gerenciado**. Processos são caracterizados por projeto e as ações são reativas.
- **Nível 3 - Definido**. Processos são caracterizados para a organização e são proativos.
- **Nível 4 - Quantitativamente gerenciado**. Os processos são medidos e controlados.
- **Nível 5 - Otimizado**. Foca na melhoria dos processos.

3.2.3 Melhoria do Processo de Software Brasileiro (MPS.br)

É um modelo de qualidade de processos criado pela Softex (Associação para Promoção da Excelência do Software Brasileiro). A implantação do modelo MPS.br tem como principal benefício o melhoramento na qualidade dos produtos aumentando assim a competitividade (SOFTEX, 2016). O objetivo do programa MPS.br é promover a melhoria de desenvolvimento de software nas empresas brasileiras.

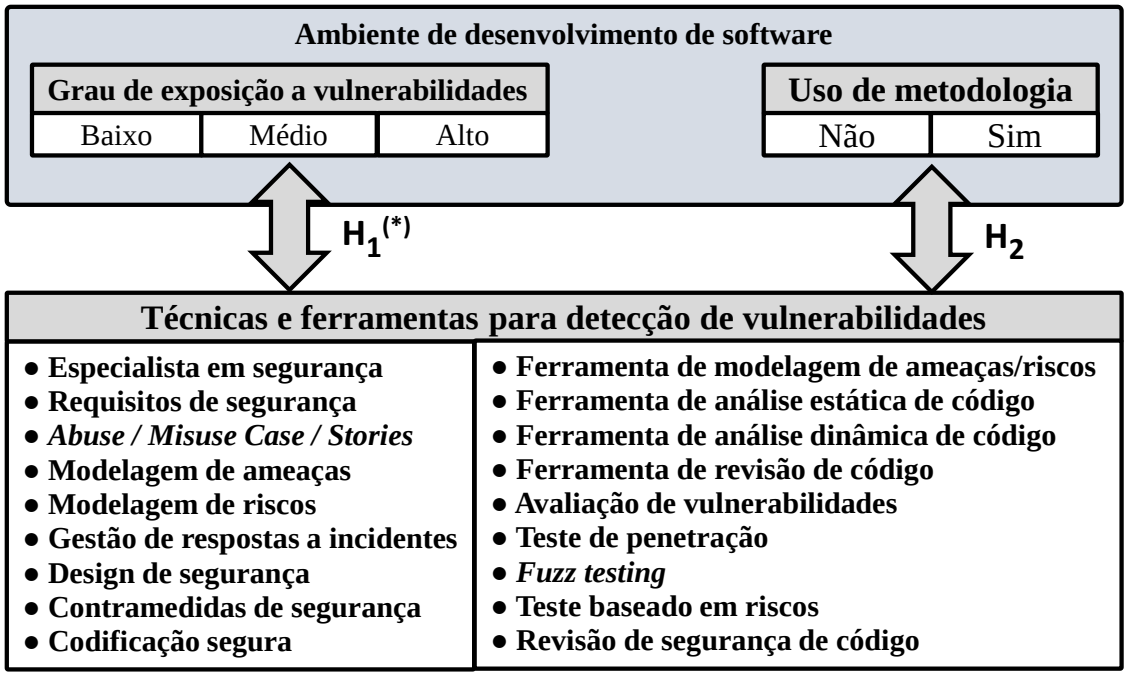
De acordo com a Softex (2016), ele possui duas metas a alcançar a médio e longo prazos: meta técnica, que visa à criação e aprimoramento do Modelo MPS.br com resultados esperados; e meta de negócio, que visa à disseminação e à adoção do modelo em todas as regiões do país, em um intervalo de tempo determinado, a um custo razoável, tanto em micro, pequenas e médias empresas.

A Softex (2016) define cinco componentes que compõem o modelo MPS.br, são eles: Modelo de Referência MPS para Software, Modelo de Referência MPS para Serviços, Modelo de Referência MPS para Gestão de Pessoas, Método de Avaliação e Modelo de Negócio para Melhoria de Processo de Software e Serviços

4 MODELO DE PESQUISA

O modelo de referência para análise da relação entre o ambiente desenvolvimento de software com o uso de TFDV é apresentado na Figura 1 e foi concebido para atender ao objetivo desta pesquisa. As variáveis presentes no modelo de pesquisa foram criadas a partir da revisão bibliográfica citada na Seção 3 deste trabalho. O modelo é composto por duas variáveis:

Figura 1. Modelo de referência da pesquisa



(*) H_1 e H_2 são hipóteses da pesquisa

Fonte: o próprio autor

(1) **Ambiente de desenvolvimento de software**. Esta variável é composta por dois indicadores.

• **Exposição a vulnerabilidades**. O grau de exposição a vulnerabilidades foi classificado em três níveis: (i) nível baixo de exposição, que representa aplicações que operam em ambientes organizacionais com ou sem acesso à web; (ii) nível médio de exposição, que representa aplicações que operam em ambientes com acesso à web e com a utilização de

recursos mobile; e (iii) nível alto de exposição, que representa aplicações que operam em ambientes com acesso à web, com a utilização de recursos mobile, e que que contemplam funcionalidade relacionadas a e-commerce. Esta classificação da exposição a vulnerabilidades considerou que ambientes com acesso à web possuem riscos devido a vulnerabilidades com maior impacto do que ambientes sem acesso à web. A utilização de recursos mobile amplifica o impacto dos riscos. Da mesma forma, o uso das aplicações em e-commerce amplifica ainda mais o impacto dos riscos.

• **Uso de modelos de qualidade de software e serviços**. Foram consideradas duas categorias: organizações que fazem uso de modelos de qualidade de software se serviços em ambientes ágeis e as organizações que não fazem uso.

(2) **Técnicas para Prevenção e Detecção de Vulnerabilidades**. Trata dos 18 tipos de TFDVs encontradas na literatura.

Com base no modelo de pesquisa foram enunciadas as seguintes hipóteses:

(H1) Associa o grau de exposição de vulnerabilidades do ambiente de desenvolvimento de software com o uso de TFDV e apresenta o seguinte enunciado: “ambientes de desenvolvimento de software com alto grau de exposição a vulnerabilidades estão associados a uma maior adoção de TFDV”.

(H2) Associa o uso de modelos, com o uso de TFDV e apresenta o seguinte enunciado: “organizações que adotam modelos de qualidade de software e serviços em ambientes ágeis estão associadas a uma maior adoção de TFDV”.

Esta hipótese foi formulada com base na premissa de que ambientes tecnologicamente complexos, ou seja, ambientes que fazem uso de técnicas de desenvolvimento ágil de software, que passam por grande pressão por entregas frequentes de software utilizável, pela competitividade no mercado, as indefinições sobre o escopo do projeto, sobre as pessoas que compõem o time, a tecnologia a ser adotada e aos riscos inerentes aos projetos, têm maior preocupação com questões de vulnerabilidades.

5 MÉTODO DE PESQUISA

Esta seção apresenta os procedimentos metodológicos que foram aplicados à pesquisa. A pesquisa proposta neste trabalho se caracteriza por ser um estudo descritivo (WRIGHTMAN; COOK; SELLTIZ, 1976). Este tipo de estudo tem como objetivo verificar a relação entre

determinadas variáveis. Trata-se de uma pesquisa quantitativa e transversal do tipo único, pois a coleta de informação foi feita uma única vez

5.1 FASES DA PESQUISA

Esta pesquisa foi desenvolvida em quatro fases. A primeira fase constituiu o levantamento bibliográfico apresentado na Seção 3. Esse levantamento descreveu os métodos ágeis e as vulnerabilidades no desenvolvimento de software.

A segunda fase definiu um modelo de referência da pesquisa para análise da relação entre ambiente de desenvolvimento de software e uso de TFDV. A construção desse modelo partiu dos conceitos apresentados na revisão bibliográfica e o modelo está apresentado na Seção 4.

Na terceira fase, foi realizado um *survey* com profissionais que atuam com desenvolvimento ágil de software. Para a realização do *survey* escolheram-se membros de equipes de desenvolvimento de software que utilizam métodos ágeis. Os respondentes ocupavam funções de analista de requisitos, arquiteto de software, desenvolvedor, testador, entre outras atribuições técnicas.

Na quarta fase, os dados coletados foram analisados, e os resultados e as conclusões da pesquisa estão apresentados nas Seções 5 e 6, respectivamente. O instrumento completo pode ser encontrado em (SANTOS; CHAIM; PRADO, 2018).

5.2 POPULAÇÃO E AMOSTRA

A unidade de análise desta pesquisa é a TFDV. As unidades de observação são os membros das equipes de desenvolvimento ágil de software pesquisados e o escopo da pesquisa abrangeu organizações brasileiras em que as equipes adotassem métodos ágeis. Os membros das equipes foram selecionados por meio da rede de profissionais *Linkedin*. Optou-se por uma amostra não probabilística, com procedimento de amostragem por conveniência, pois essas características são adequadas para a obtenção de informações com custo menor (SOMMERVILLE, 2011).

5.3 COLETA E ANÁLISE DE DADOS

Os dados coletados são do tipo primário, ou seja, são aqueles que não foram antes coletados. Adotou-se nesta pesquisa o questionário estruturado como instrumento de coleta de dados. A vantagem desse instrumento está no custo de aplicação e na uniformidade de

mensuração. Além de ser a melhor forma de coletar informações de muitos respondentes (AAKER; KUMAR; DAY, 2004). Os dados foram coletados no segundo semestre de 2017.

5.4 ANÁLISE E TRATAMENTO DOS DADOS

A análise de dados foi realizada em duas etapas. Na primeira etapa foram utilizadas estatísticas descritivas. O objetivo dessa etapa foi descrever as características da amostra. Na segunda e última etapa utilizou-se a medida de associação Gama, voltada para associação entre variáveis ordinais, de acordo com Kinnear e Taylor (1979). Esta medida de associação está apresentada a seguir:

Medida de Associação Gama (γ). A estatística Gama não só indica a força de associação entre as variáveis, mas também sua direção. Sua mensuração baseia-se no cálculo de pares concordantes e discordantes para todos os pares de observação das variáveis. O cálculo é feito através da seguinte fórmula:

$$\gamma = \frac{(P - Q)}{(P + Q)}$$

Onde: P = número de pares concordantes

Q = número de pares discordantes

Para auxiliar a interpretação dessa estatística, Babbie *et al*. (2000) associaram intervalos de valores a uma medida de força da associação, conforme indicado na Tabela 2.

Tabela 2. Interpretação das medidas de associação Lambda

Força da associação	**Medida da associação Gama**	
Nenhuma		0,00
Fraca	± 0,01 a	0,09
Moderada	± 0,10 a	0,29
Forte	± 0,30 a	0,99
Perfeita		1,00

Fonte: adaptado de Babbie *et al*. (2000)

Brazilian Journal of Development

6 APRESENTAÇÃO DOS RESULTADOS

A análise dos dados e os resultados estão apresentados em quatro tópicos: (1) Características dos Respondentes; (2) Características do Ambiente; (3) Resultado das Hipóteses de Pesquisa; e (4) Análise dos Resultados. A amostra obtida foi de 76 profissionais da área de desenvolvimento ágil de software.

6.1 CARACTERÍSTICAS DOS RESPONDENTES

As características da amostra foram analisadas em relação aos respondentes e ao ambiente de desenvolvimento de software. As características dos respondentes estão apresentadas na Tabela 3. Em relação à função exercida pelos respondentes, a maioria é representada por desenvolvedores e *agile coaches*, que representam mais da metade da amostra (69,7% = 42,1%+27,6%).

Tabela 3. Características do respondente e da equipe

Variável	**Categoria ou níveis**	**Frequência**	
		Absoluta	**Relativa**
Função	Desenvolvedor	32	42,1
	Agile Coach	21	27,6
	Testador	8	10,5
	Especialista em segurança	6	7,9
	Analista de requisitos	3	3,9
	Arquiteto de software	2	2,6
	Outros	4	5,3
	Total	**76**	**100,0**
Tamanho da equipe	0 a 5	39	51,3
	6 a 10	18	23,7
	Acima de 10	19	25,0
	Total	**76**	**100,0**
Tempo na função	0 a 5	39	51,3
	6 a 10	31	40,8
	Acima de 10	6	7,9
	Total	**76**	**100,0**

Fonte: próprio autor

As equipes são na maioria pequenas (51,3%), ou seja, com até cinco integrantes. Isso é característico de equipes que adotam métodos ágeis. Por último, destaca-se a experiência na função. A maioria (51,3%) possui pouca experiência (até 5 anos) e apenas 7,9% da amostra possui muita experiência na função.

Brazilian Journal of Development

6.2 CARACTERÍSTICAS DO AMBIENTE

As características do ambiente de desenvolvimento ágil de software estão apresentadas na Tabela 4. Em relação ao uso de modelos de qualidade de software e serviços, a maioria (59,2% = 45/76) das organizações pesquisadas usam modelos próprios ou proveniente do mercado de tecnologia da informação. Por outro lado, em relação aos ambientes tecnológicos em que operam, aplicações web e aplicações que possuem recursos mobile e de e-commerce foram a maioria com 76,3% da amostra.

Tabela 4. Características do ambiente de desenvolvimento

Ambiente de desenvolvimento de software		**Usa Metodologia**		**Total**
		Não	**Sim**	
Grau de exposição a vulnerabilidades	Baixo	19	19	38
	Médio	6	17	23
	Alto	6	9	15
Total		**31**	**45**	**76**

Fonte: próprio autor

6.3 RESULTADO DAS HIPÓTESES DE PESQUISA

As hipóteses H1 e H2 foram verificadas pela aplicação da técnica estatísticas *Fisher´s Exact Test* (FISHER; BENNETT, 1990), e só foram consideradas as relações com nível de significância estatística menor ou igual a 5%. Os resultados da análise estão sumarizados na Tabela 5.

A hipótese H1 associa o grau de exposição de vulnerabilidades do ambiente de desenvolvimento de software com o uso de TFDV. Oito das 18 TFDV tiveram forte associação (maior que 0,30) com a exposição de vulnerabilidades do ambiente de desenvolvimento ágil de software. Entretanto, todas as associações foram negativas, indicando uma relação inversa, ou seja, quanto mais complexo o ambiente de desenvolvimento de software, menor é a adoção de TFDV.

A hipótese H2 associa o uso de modelos de qualidade de software e serviços com o uso de TFDV. Sete das 18 TFDV tiveram forte associação (maior que 0,30). Entretanto, quatro das sete associações foram positivas, indicando uma relação direta, ou seja, o uso de modelos

de qualidade de software e serviços está positivamente associado a uma maior a adoção de TFDV.

Tabela 5. Medidas de associação Gama

Técnicas e Ferramentas de Detecção de Vulnerabilidades	**Exposição a vulnerabilidades**	**Uso de metodologias**
Especialista em segurança		-0,372
Requisitos de segurança		
Abuse/misuse Case/Stories		0,442
Modelagem de ameaças	-0,485	
Modelagem de riscos	-0,724	0,309
Gestão de respostas a incidentes	-0,037	
Design de segurança		
Contramedidas de segurança		
Codificação segura		0,329
Ferramenta de modelagem de ameaças/riscos	-0,568	-0,337
Ferramenta de análise estatística de código	-0,342	
Ferramenta de análise dinâmica de código		
Ferramenta de revisão de código		
Avaliação de vulnerabilidades		0,441
Teste de penetração	-0,437	
Fuzz testing	-0,449	-0,694
Testes baseados em riscos	-0,462	
Revisão de segurança de código		

Fonte: próprio autor

6.4 ANÁLISE DOS RESULTADOS

Em relação à função exercida pelos respondentes, a maioria é representada por desenvolvedores de software e *agile coaches*, e eles representam mais da metade da amostra. As demais funções foram pouco representativas na amostra pelo fato de o *survey* ter como contexto os métodos ágeis. As equipes de desenvolvimento de software são compostas por cinco a seis integrantes. Esta quantidade tem por objetivo minimizar a complexidade na organização da equipe e quanto aos aspectos de gerenciamento e entendimento do processo ágil. Por último, destaca-se a experiência na função desempenhada atualmente na equipe. A maioria dos profissionais da amostra possui pouca experiência na função atual.

São desenvolvedores, *scrum masters* e *agile coaches* com no máximo 5 anos de experiência. Em relação ao uso de modelos de qualidade de software e serviços que contribuem na incorporação de TFDVs, a maioria faz uso de metodologias próprias ou proveniente do mercado de tecnologia da informação.

Por outro lado, em relação aos ambientes tecnológicos para os quais as organizações desenvolvem soluções, as aplicações web, mobile e de e-commerce foram a maioria na amostra. Isso pode ser explicado pelo fato de cada vez mais as organizações investirem em inovação tecnológica como, por exemplo, o uso de técnicas de inteligência artificial, internet das coisas entre outras.

Quanto à hipótese H1, ela tem por objetivo associar o grau de exposição de vulnerabilidades do ambiente de desenvolvimento de software com o uso de TFDV. Como foi apresentado anteriormente, oito das 18 TFDVs tiveram forte associação com a exposição de vulnerabilidades do ambiente de desenvolvimento de software. Entretanto, todas as associações foram negativas, indicando uma relação inversa, ou seja, quanto mais complexo o ambiente de desenvolvimento ágil de software, menor é a adoção de TFDVs.

Uma possível explicação é que para uma TFDV ser incorporada em um processo ágil, é necessário que um especialista incentive esta ação. No entanto, são poucas as equipes que incluem um profissional de segurança. Portanto, técnicas de segurança precisam ser incorporadas na cultura organizacional e consequentemente da equipe. Esse objetivo pode ser atingido com técnicas de baixa tecnologia (*low tech*) como estórias de usuários voltadas para segurança, *misuse cases* e *abuse cases*.

Já quanto à hipótese H2, quatro das sete associações foram positivas, ou seja, o uso de modelos de qualidade de software e serviços como CMMI, ITIL, MPS.br está positivamente associado a uma maior a adoção de TFDV. Isso indica que os modelos de qualidade de software e serviços apoiam e possuem aderência quando existe a necessidade de incorporar TFDVs aos processos ágeis.

Estes modelos contribuem para o uso das melhores práticas e gerenciamento de serviços de TI, qualidade e processos. Logo, os resultados indicam que elas criam um ambiente favorável para o uso de TFDVs, no qual estas podem ser potencializadas.

7 CONCLUSÕES

O objetivo deste artigo foi descrever o uso de técnicas e ferramentas de detecção de vulnerabilidades TFDV em ambientes de desenvolvimento ágil de software, com o uso de modelos de qualidade de software e serviços. Para isso foram elaboradas duas hipóteses voltadas para a utilização de TFDVs: a primeira abordou a aplicação de TFDVs em ambientes

de desenvolvimento ágil de software e a segunda tratou dos modelos de qualidade de software e serviços que são utilizadas juntamente com TFDVs.

A partir dos dados obtidos da aplicação do survey, considerando a primeira hipótese, foi possível afirmar que em ambientes complexos ocorre pouca utilização de TFDV. Assim, é necessário identificar quais TFDVs podem ser aplicadas em ambientes com constantes mudanças, sem prejudicar ou influenciar negativamente os processos. Ou seja, independentemente do ambiente e estado em que o projeto se encontra, deve ser possível fazer uso de TFDVs para que este processo se torne o mais natural possível e sem que o ágil perca a sua essência. Já quanto aos modelos de qualidade e serviços, eles apoiam e possuem aderência ao incorporar TFDVs nos processos ágeis, pois atuam no uso das melhores práticas e gerenciamento de serviços de TI, qualidade e processos. Com o uso desses modelos, as TFDVs podem ser potencializadas sem prejudicar os processos já existentes.

Com este estudo foi possível concluir que em ambientes de desenvolvimento ágil de software que passam por grande pressão por entregas frequentes de software utilizável, competitividade no mercado, indefinições sobre o escopo do projeto, sobre as pessoas que compõem o time, a tecnologia a ser adotada e aos riscos inerentes aos projetos, é pouca a preocupação com segurança de software.

Isso pode ser motivado pela falta de conhecimento, pela falta de interesse ou pela falta de oportunidade em entender e aplicar segurança, visto a complexidade do ambiente e pressões internas e externas. Por outro, os modelos de qualidade de software e serviços favorecem a adoção de TFDVs por apoiarem a adoção de melhores práticas.

Esta pesquisa possui limitações das quais se destaca o fato de a amostra não ser aleatória e, portanto, os resultados não podem ser generalizados. Esta pesquisa faz parte de um projeto iniciado em 2016 com o objetivo de analisar as relações entre o uso de TFDV, métodos ágeis e modelos de qualidade de software e serviços. Os resultados desta pesquisa corroboram as relações sugeridas.

Assim, sugerem-se futuras pesquisas que busquem confirmar, por meio de pesquisa explanatória, as relações aqui identificadas.

Brazilian Journal of Development

REFERÊNCIAS

AAKER, D. A.; KUMAR, V; DAY, G. S. **Marketing research**, 7th edition. John Wiley's & Sons, New York, USA, 2004.

AXELOS. **IT Service Management. What is ITIL?**, 2018. Disponível em: <https://www.axelos.com/best-practicesolutions/itil/what-is-it-service-management >.

BABBIE, E.; HALLEY, E.; ZANINO, F. J. **Adventures in Social Research**. California: Sage Publications, 2000.

BACA, D.; BOLDT, M.; CARLSSON, B.; JACOBSSON, A. A Novel Security-Enhanced Agile Software Development Process Applied in an Industrial Setting. In **10th International Conference on Availability, Reliability and Security**, p. 11-19, 2015.

BARTSCH S. Practitioners' perspectives on security in agile development. In **Sixth International Conference on Availability, Reliability and Security**, p. 479-484, 2011.

BECK, K. **Extreme programming explained: Embrace change**, 2nd edition. Addison - Wesley, Upper Saddle River, NJ, 2000.

BISHOP, M. **Computer security: art and science**. Addison- Wesley Professional, 2003.

CERT.br. Centro de Estudos, Resposta e Tratamento a Incidentes de Segurança no Brasil. **Cartilha de segurança para internet: ataques na internet**, 2012. Disponível em: <https://cartilha.cert.br/>.

CMMI. **IT What Is Capability Maturity Model Integration (CMMI)?**, 2018. Disponível em:<https://cmmiinstitute.com/>.

CRISTOFOLI, F.; PRADO, E. P. V.; TAKAOKA H. Gestão da Terceirização da Tecnologia da Informação Baseada nas Práticas de Governança In **International Conference on Information Systems and Technology Management**. 2012.

CRUZES, D. S.; FELDERER, M.; OYETOYAN, T. D.; GANDER, T. M.; PEKARIC, I. How is security testing done in agile teams? A cross-case analysis of four software teams. In **International Conference on Agile Software Development**, p. 201–216, 2017.

FISHER, R. A.; BENNETT, J. H. Statistical methods, experimental design, and scientific inference, 1990.

GREGOIRE, J. et al. On the secure software development process: CLASP and SDL compared. In **Proceedings of the Third International Workshop on Software Engineering for Secure Systems**, p. 1-7, 2007.

GROUP, I. IEEE Standard classification for software anomalies. In **IEEE Standards Associations**, p. 1-24, 2010.

GTIESKAMP, W.; GUREVICH, Y.; SCHULTE, W.; VEANES, M. Generating finite state machines. In **ACM SIGSOFT**, p. 112-122, 2002.

HOWARD, M.; LIPNER, S. **The security development lifecycle**, 1st edition. Microsoft Press Redmond, WA, USA, 2006.

KHAIM, R.; NAZ, S.; ABBAS, F.; IQBA, N.; HAMAYUN, M.; PAKISTAN, R. A review of security integration technique in agile software development. **International Journal of Software Engineering & Applications**, v. 7, n. 3, 2016.

KINNEAR, T. C.; TAYLOR, J. R. **Marketing Research: an applied approach**. International Student Edition, McGraw- Hill, Tokyo, 1979.

MÁRQUEZ, P. G.; SILVIA, R.; NOEL, S.; MATALONGA; ASTUDILLO, H. Identifying emerging security concepts using software artifacts through an experimental case. In **Chilean Computer Science Society (SCCC)**, p. 1-6, 2015.

MCGRAW, G. **Software security: building security in**, 1st edition. Addison- Wesley Professional, 2006.

MCGRAW, G.; MORRISSETT, G. Attacking malicious code. In **IEEE Computer Society**, p. 1-11, 2000.

NIST. National Institute of Standards and Technology. **National Vulnerability Database – NVD**, 2009. Disponível em: <https://nvd.nist.gov/vuln>.

OTHMANE, L. B.; ANGIN, P.; WEFFERS, H.; BHARGAVA, B. Extending the agile development process to develop acceptably secure software. In **IEEE Transaction on Dependable and Secure Computing**, v. 11, n. 6, p. 497-509, 2014.

OWASP. **OWASP CLASP Concepts**, 2016. Disponível em: <https://www.owasp.org/index.php/CLASP_Concepts>.

OWASP. **OWASP Top Ten**, 2017. Disponível em: <https://www.owasp.org/index.php/Top_10-2017_Top_10>.

OYETOYAN, T. D.; CRUZES, D. S.; GILJE, M. J. An empirical study on the relationship between software security skills, usage, and training needs in agile settings. In **11th International Conference on Availability, Reliability and Security**, 2016.

PINHEIRO, J. M. S. Ameaças e ataques aos sistemas de informação: Prevenir e antecipar. **UniFOA**, p. 1-11, 2011.

SANTOS, L. C. M. C.; CHAIM, M. L.; PRADO, E. P. V., 2018. **Instrumento do Survey sobre Técnicas e Ferramentas de Detecção de Vulnerabilidades**. Disponível em: <https://github.com/SAEG1/InstrumentoEstudodeCaso.git/>.

SEACORD, R. C.; HOUSEHOLDER, A. D. **A Structured approach to classifying security vulnerabilities**. CMU SEI, 2005.

Softex. **Guia Geral de Software**, 2016. Disponível em: <https://www.softex.br/mpsbr/guias/>.

SOMMERVILLE, I. **Software Engineering**, 9th edition. Pearson Education, Boston, Massachusetts, 2011.

STALLINGS. **Computer security**. Pearson Education, Boston, Massachusetts, 2nd edition, 2012.

VERDON, D.; MCGRAW, G. Risk analysis. **IEEE Security & Privacy**, p. 79-84, 2004.

VIANNA, E. W.; FERNANDES, J. H. C. O gestor da segurança da informação no espaço cibernético governamental: Grandes desafios, novos perfis e procedimentos. **Brazilian Journal of Information Science**, v. 9, n. 1, p. 1-28, 2015.

WEBER, T. S. Tolerância a falhas: Conceitos e exemplos. In: **Programa de Pós-Graduação– Instituto de Informática- UFRGS**, p. 1-5, 2003.

WRIGHTMAN, L. S.; COOK, S. W.; SELLTIZ, C. **Research Methods in Social Relations**, 3rd edition. Holt, Rinehart Winston, New York, 1976.