UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

LÍGIA CÁSSIA MORENO DE CASTRO SANTOS

**Uso de técnicas e ferramentas para detecção de vulnerabilidades: um *survey*
com membros de equipes de desenvolvimento ágil de software**

São Paulo

2018

LÍGIA CÁSSIA MORENO DE CASTRO SANTOS

**Uso de técnicas e ferramentas para detecção de vulnerabilidades: um *survey*
com membros de equipes de desenvolvimento ágil de software**

Versão corrigida

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação.

Área de concentração: Metodologia e Técnicas da Computação

Versão corrigida contendo as alterações solicitadas pela comissão julgadora em 05 de Abril de 2018. A versão original encontra-se em acervo reservado na Biblioteca da EACH-USP e na Biblioteca Digital de Teses e Dissertações da USP (BDTD), de acordo com a Resolução CoPGr 6018, de 13 de outubro de 2011.

Orientador: Prof. Dr. Marcos Lordello Chaim

São Paulo

2018

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

CATALOGAÇÃO-NA-PUBLICAÇÃO

(Universidade de São Paulo. Escola de Artes, Ciências e Humanidades. Biblioteca)

CRB-8 4936

Santos, Lígia Cássia Moreno de Castro

Uso de técnicas e ferramentas para detecção de vulnerabilidades: um *survey* com membros de equipes de desenvolvimento ágil de software / Lígia Cássia Moreno de Castro Santos ; orientador, Marcos Lordello Chaim. – 2018.

162 f. : il

Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Sistemas de Informação, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo.

Versão corrigida

1. Teste e avaliação de software. 2. Desenvolvimento de software. 3. Engenharia de software. I. Chaim, Marcos Lordello, orient. II. Título

CDD 22.ed.– 005.14

Dissertação de autoria de Lígia Cássia Moreno de Castro Santos, sob o título “**Uso de técnicas e ferramentas para detecção de vulnerabilidades: um *survey* com membros de equipes de desenvolvimento ágil de software**”, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Sistemas de Informação, na área de concentração Metodologia e Técnicas da Computação, aprovada em 05 de Abril de 2018 pela comissão julgadora constituída pelos doutores:

Prof. Dr. Marcos Lordello Chaim

Presidente
Instituição: EACH-USP

Prof. Dra. Ellen Francine Barbosa

Instituição: ICMC-USP

Prof. Dra. Sarajane Marques Peres

Instituição: EACH-USP

Prof. Dra. Daniela Soares Cruzes

Instituição: SINTEF

Dedico esta dissertação aos meus familiares e a Deus.

Agradecimentos

Primeiramente agradeço a Deus por permitir que eu realizasse esta etapa com saúde, perseverança e sabedoria.

Agradeço aos meus familiares por estarem ao meu lado sempre, pelo incentivo, apoio, confiança e compreensão.

Ao meu orientador prof. Dr. Marcos Lordello Chaim, por todas as orientações, os ensinamentos constantes, as oportunidades, a disponibilidade, a paciência e o crescimento que me proporcionou.

Ao prof. Dr. Edmir Parada Vasques Prado pela grande colaboração em meu projeto, pelos ensinamentos constantes, a disponibilidade em me atender frequentemente e a paciência em passar seu conhecimento.

Aos meus amigos do laboratório SAEG/GeSI e GrIA que proporcionaram bons momentos de descontração e me acalmaram nos momentos de nervosismo.

Em especial a Carolina Luiza Chamas pelos grupos de estudos que organizávamos para as disciplinas do programa e pela disponibilidade e ao Jonas Mendonça Targino por ter paciência comigo, por me auxiliar e me proporcionar risadas nos momentos em que o choro era certo.

Agradeço a oportunidade em ter sido monitora PAE e adquirido conhecimento em sala de aula nas disciplinas ministradas pelos professores Dr. Marcelo Fantinato e Dr. Daniel Cordeiro.

Aos professores da Faculdade de Tecnologia da Zona Leste (FATEC-ZL), em especial a prof. Dra. Cristina Correa de Oliveira, minha ex-orientadora. Agradeço pela confiança em ter a mim como monitora da disciplina de Engenharia de Software por 1 ano e por enviar a carta de recomendação para que eu entrasse no mestrado.

Agradeço aos demais professores e colegas que contribuíram de alguma forma para a conclusão desta dissertação.

*“Os sonhos não determinam o lugar em que você vai estar, mas produzem a força
necessária para tirá-lo do lugar em que está.”*
(Augusto Cury)

Resumo

SANTOS, Lígia Cássia Moreno de Castro. **Uso de técnicas e ferramentas para detecção de vulnerabilidades:** um *survey* com membros de equipes de desenvolvimento ágil de software. 2018. 162 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2018.

Métodos ágeis foram criados para sanar fraquezas reais e perceptíveis dos métodos tradicionais de desenvolvimento de software. Devido à pressão na entrega de produtos de software dentro do prazo, muitas vezes requisitos de segurança são pouco mensurados ou até deixados de lado. Durante o desenvolvimento ágil de software é importante detectar possíveis vulnerabilidades. Esta dissertação descreve um *survey* aplicado a membros de equipes de desenvolvimento de software que aplicam métodos ágeis. Para tanto, foram identificados por meio da rede de profissionais LinkedIn 110 membros de equipes ágeis que implantaram, estão em processo de implantação ou ainda irão implantar técnicas e ferramentas para detecção de vulnerabilidades. Além disso, foram entrevistados nove gerentes de equipes ágeis. O questionário e o roteiro da entrevista foi baseado em três conhecidos processos de desenvolvimento de software seguro, a saber, Processo de McGraw, OWASP CLASP e as atividades de Howard e Lipner. A coleta de dados se deu por meio de questionários e entrevistas. A análise dos resultados utilizou técnicas de estatística descritiva e análise de conteúdo. Elas indicaram os métodos ágeis mais utilizados, o uso atual das técnicas e ferramentas, as aptidões, os interesses e as necessidades em treinamento em técnicas e ferramenta para detecção de vulnerabilidades. Além disso, os benefícios obtidos com a implantação das técnicas e ferramentas, as motivações, as estratégias, as dificuldades, as limitações e as lições aprendidas foram identificadas. Os resultados indicam que existe motivação para a implantação de segurança, mas ainda não se dá atenção especial à detecção de vulnerabilidades nas equipes ágeis cujos membros participaram do *survey*.

Palavras-chaves: Desenvolvimento de software. Engenharia de Software. Métodos Ágeis. Segurança de Software. Vulnerabilidades. Técnicas e ferramentas para detecção de vulnerabilidades. Survey.

Abstract

SANTOS, Lúcia Cássia Moreno de Castro. **Use of techniques and tools for vulnerability detection:** a survey with members of agile software development teams. 2018. 162 p. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2018.

Agile methods were created to address real and perceived weaknesses of traditional software development methods. Due to the pressure to delivery software products on time, security requirements are often poorly addressed or even neglected. During agile software development it is important to detect possible vulnerabilities. This dissertation describes a survey applied to members of software development teams who apply agile methods. Thus, 110 members of agile teams were identified through LinkedIn's network of professionals who deployed, are in the process of being deployed or will still implement techniques and tools for vulnerability detection techniques and tools were identified. The questionnaire was based on three known safe software development processes, namely, the McGraw Process, OWASP CLASP, and the activities of Howard and Lipner. Data were collected through questionnaires and interviews. The analysis of the results used techniques of descriptive statistics and content analysis. They indicated the most widely used agile methods, the current use of techniques and tools, the skills, interests and training needs of agile teams in vulnerability detection techniques and tools. In addition, the benefits of implementing the techniques and tools, the motivations, the strategies, the difficulties, the limitations and the lessons learned were identified. The results suggest that special attention is still not given to detection of vulnerabilities in the agile teams whose members participated in the survey.

Keywords: Software Development. Software Engineering. Agile Methods. Software Security. Vulnerabilities. Techniques and tools for vulnerabilities detection. Survey.

Lista de figuras

Figura 1 – Exemplo de um <i>misuse case</i> representando um <i>mis</i> -ator realizando tentativas de ataques a um sistema de autenticação de um ator	40
Figura 2 – Exemplo de um <i>abuse case</i> que representa um ator malicioso realizando tentativas de ataques a um sistema de login de usuários.	41
Figura 3 – Exemplo de uma árvore de ameaças e suas mitigações	43
Figura 4 – Artefatos produzidos em cada fase do ciclo de vida de desenvolvimento de software seguro do processo de McGraw	52
Figura 5 – Modelo do processo de coleta de dados em um <i>survey</i> utilizando questionário	68
Figura 6 – Iterações e uso das técnicas e ferramentas nas fases do ciclo de vida ágil	72

Lista de quadros

Quadro 1 – Comparação entre os processos de desenvolvimento de software seguro.	58
Quadro 2 – Cadeia de caracteres	61
Quadro 3 – Trabalhos relacionados	61
Quadro 4 – Matriz de amarração	69
Quadro 5 – Técnicas e ferramentas para detecção de vulnerabilidades utilizadas pelos processos de segurança	71
Quadro 6 – Escala de níveis de habilidades nas técnicas e ferramentas de segurança de software	73
Quadro 7 – Exemplos de técnicas e ferramentas	154

Lista de tabelas

Tabela 1 – Função dos respondentes e anos de atuação	78
Tabela 2 – Métodos ágeis utilizados, tecnologias adotadas, metodologias, equipe e responsável pela implantação de segurança	80
Tabela 3 – Uso atual de técnicas e ferramentas para detecção de vulnerabilidades nos projetos	83
Tabela 4 – Grau de implantação e ocorrência de benefícios na implantação das técnicas e ferramentas para detecção de vulnerabilidades	86
Tabela 5 – Grau de implantação e ocorrência de benefícios na implantação das técnicas e ferramentas para detecção de vulnerabilidades por fase . . .	89
Tabela 6 – Aptidão dos membros nas técnicas e ferramentas	92
Tabela 7 – Interesse da equipe de desenvolvimento em técnicas e ferramentas para detecção de vulnerabilidades	94
Tabela 8 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades	95
Tabela 9 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades	97
Tabela 10 – Motivação para a implantação de segurança no desenvolvimento ágil de software	98
Tabela 11 – Porquês das escolhas das técnicas e ferramentas para detecção de vulnerabilidades	99
Tabela 12 – Detalhes da implantação das técnicas e ferramentas para detecção de vulnerabilidades	100
Tabela 13 – Estratégias utilizadas para a implantação dessas técnicas e ferramentas para detecção de vulnerabilidades	101
Tabela 14 – Dificuldades e limitações encontradas na implantação de atividades de segurança	102
Tabela 15 – Lições aprendidas com a implantação de técnicas e ferramentas para detecção de vulnerabilidades	104
Tabela 16 – Função dos respondentes e anos de atuação	105

Tabela 17 – Métodos ágeis utilizados, tecnologias adotadas e composição da equipe e responsável pela implantação de segurança	106
Tabela 18 – Uso atual de técnicas e ferramentas para detecção de vulnerabilidades nos projetos pelos gerentes	108
Tabela 19 – Aptidão dos gerentes em técnicas e ferramentas para detecção de vulnerabilidades	112
Tabela 20 – Interesse dos gerentes nas técnicas e ferramentas para detecção de vulnerabilidades	115
Tabela 21 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades	117
Tabela 22 – Grau de interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades	118
Tabela 23 – Motivação para a implantação de segurança no desenvolvimento ágil de software	119
Tabela 24 – Escolha das técnicas e ferramentas para detecção de vulnerabilidades .	120
Tabela 25 – Detalhes sobre a implantação das técnicas e ferramentas para detecção de vulnerabilidades	121
Tabela 26 – Estratégias utilizadas para a implantação das técnicas e ferramentas para detecção de vulnerabilidades	122
Tabela 27 – Dificuldades e limitações encontradas na implantação	122
Tabela 28 – Lições aprendidas com a implantação das técnicas e ferramentas para detecção de vulnerabilidades	124
Tabela 29 – Grau de implantação e benefícios obtidos com a implantação das técnicas e ferramentas	138
Tabela 30 – Motivação da implantação	155
Tabela 31 – Escolhas das técnicas e ferramentas para detecção de vulnerabilidades .	156
Tabela 32 – Detalhes da implantação das técnicas e ferramentas	157
Tabela 33 – Estratégias utilizadas para a implantação das técnicas e ferramentas para detecção de vulnerabilidades	158
Tabela 34 – Dificuldades e limitações encontradas na implantação de atividades de segurança	160

Tabela 35 – Lições aprendidas com a implantação de técnicas e ferramentas para
detecção de vulnerabilidades 161

Sumário

1	Introdução	18
1.1	<i>Motivação</i>	19
1.2	<i>Justificativa</i>	19
1.3	<i>Objetivos</i>	20
1.4	<i>Resultados obtidos</i>	21
1.5	<i>Estrutura da dissertação</i>	21
2	Métodos ágeis	23
2.1	<i>História</i>	23
2.2	<i>Valores e princípios dos métodos ágeis</i>	24
2.3	<i>Principais métodos ágeis</i>	26
2.3.1	Programação extrema - XP	27
2.3.2	Scrum	30
2.4	<i>Aplicação dos métodos ágeis</i>	32
2.5	<i>Considerações finais</i>	34
3	Vulnerabilidades	35
3.1	<i>Principais vulnerabilidades de software</i>	36
3.2	<i>Outras vulnerabilidades</i>	38
3.3	<i>Técnicas e ferramentas para detecção de vulnerabilidades</i>	38
3.3.1	Análise de requisitos e modelagem	38
3.3.2	<i>Design</i>	43
3.3.3	<i>Codificação</i>	44
3.3.4	<i>Testes</i>	47
3.3.5	<i>Produção</i>	49
3.4	<i>Considerações finais</i>	50
4	Processos de desenvolvimento de software seguro	51
4.1	<i>Processo de McGraw</i>	52
4.2	<i>Ciclo de Vida de Desenvolvimento de Segurança da Microsoft (MS SDL - Microsoft Security Development Life Cycle)</i>	52

4.3	<i>Modelo de Segurança no Desenvolvimento de Software (SSDM - Secure Software Development Model)</i>	53
4.4	<i>Processo Leve e Abrangente de Aplicação de Segurança (CLASP - Comprehensive, Lightweight Application Security Process)</i>	53
4.5	<i>Processo para Equipes de Desenvolvimento de Software Seguro (TSP Secure - Team Software Process for Secure Software Development)</i> . .	54
4.6	<i>Modelo de Maturidade para Construção de Segurança (BSIMM - Building Security in Maturity Model)</i>	54
4.7	<i>Métodos ágeis e segurança de software</i>	56
4.8	<i>Comparação dos principais processos</i>	57
4.9	<i>Considerações finais</i>	59
5	Trabalhos relacionados	60
5.1	<i>Métodos ágeis e detecção de vulnerabilidades</i>	60
5.2	<i>Seleção dos estudos</i>	60
5.3	<i>Descrição dos trabalhos relacionados</i>	61
5.3.1	Avaliações empíricas	62
5.3.2	Extensões de métodos ágeis e comparações entre processos	63
5.4	<i>Comparação com os trabalhos selecionados</i>	65
5.5	<i>Considerações finais</i>	66
6	Procedimentos metodológicos	67
6.1	<i>Estratégia de pesquisa</i>	67
6.2	<i>Relação entre objetivos, métodos e resultados</i>	68
6.3	<i>Survey</i>	70
6.3.1	Sujeitos da pesquisa	70
6.3.2	Instrumento do survey	70
6.3.3	Coleta de dados	74
6.3.4	Tratamento dos dados	74
6.4	<i>Considerações finais</i>	76
7	Análise dos resultados	77
7.1	<i>Resultados obtidos com o survey</i>	77

7.1.1	Características dos membros das equipes de desenvolvimento ágil de software	77
7.1.2	Características dos projetos	79
7.2	<i>Técnicas e ferramentas para detecção de vulnerabilidades</i>	82
7.2.1	Identificação e descrição das técnicas e ferramentas adotadas pelos membros de equipes ágeis	82
7.2.2	Grau de implantação das técnicas e ferramentas e benefícios associados	84
7.3	<i>Aptidão, interesse e necessidade de treinamento</i>	90
7.3.1	Aptidão dos membros de equipes ágeis em técnicas e ferramentas .	90
7.3.2	Interesse dos membros de equipes ágeis nas técnicas e ferramentas .	93
7.3.3	Necessidade de treinamento dos membros de equipes ágeis nas técnicas e ferramentas	96
7.3.4	Características da implantação das técnicas e ferramentas	98
7.4	<i>Resultado das entrevistas com os gerentes</i>	104
7.4.1	Características dos gerentes de equipes de desenvolvimento ágil de software	105
7.4.2	Características dos projetos	105
7.5	<i>Técnicas e ferramentas para detecção de vulnerabilidades</i>	107
7.5.1	Identificação e descrição das técnicas e ferramentas adotadas pelos gerentes de equipes ágeis	108
7.6	<i>Aptidão, interesses e necessidades de treinamento</i>	110
7.6.1	Aptidão dos gerentes de equipes ágeis em técnicas e ferramentas .	110
7.6.2	Interesse dos gerentes de equipes ágeis nas técnicas e ferramentas .	113
7.6.3	Necessidade de treinamento dos gerentes de equipes ágeis nas técnicas e ferramentas	117
7.6.4	Características da implantação das técnicas e ferramentas	119
7.7	<i>Ameaças à validade</i>	125
7.7.1	Considerações finais	126
8	Conclusões	127
8.1	<i>Síntese do trabalho</i>	127
8.2	<i>Possíveis contribuições</i>	129
8.3	<i>Trabalhos futuros</i>	129

Referências¹	131
Apêndice A – Instrumento do <i>survey</i>	138
A.1 <i>Instrumento aplicado aos gerentes de equipes de desenvolvimento ágil de software</i>	138
A.2 <i>Instrumento aplicado aos membros de equipes de desenvolvimento ágil de software</i>	145
A.3 <i>Roteiro semi-estruturado utilizado para condução das entrevistas</i>	151
A.4 <i>Exemplos de técnicas e ferramentas para detecção de vulnerabilidades</i>	154
Apêndice B – Mais características da aplicação das técnicas e ferramentas para detecção de vulnerabilidades	155

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

1 Introdução

Atualmente, as organizações operam em um ambiente com constantes mudanças culturais, sociais, políticas e econômicas. Com isso elas devem responder rapidamente a essas mudanças e a pressões competitivas. Processos de desenvolvimento de software que planejam especificar completamente os requisitos e, em seguida, projetar, construir e testar o sistema de forma sequencial e rígida se tornam inadequados para ambientes que sofrem constantes mudanças (SOMMERVILLE, 2011). Se os requisitos de software mudam, conseqüentemente o projeto, a implementação e os testes do sistema mudam também. Por conta disso, torna-se difícil obter um conjunto completo de requisitos estáveis de software (PRESSMAN, 2009).

Para sistemas maiores ou críticos, é necessária, porém, uma abordagem mais detalhada e rígida, conhecida também como orientada a planos, tradicional (SOMMERVILLE, 2011). No entanto, em ambientes com constantes mudanças, uma abordagem ágil é recomendável. Os métodos ágeis possuem valores e princípios que devem ser seguidos em um projeto para ser considerado ágil (BECK; BEEDLE et al., 2001).

Pela agilidade do processo de desenvolvimento e 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, 2010b). De acordo com a OWASP (2010b) a maioria das vulnerabilidades nos sistemas de software são causadas por falta de aplicação de políticas de segurança no projeto. Para evitar isso, é necessário analisar o uso de práticas de segurança pelas organizações, as habilidades dos membros e as necessidades de formação da equipe, bem como adaptar os processos de segurança de desenvolvimento de software tradicional aos processos dos métodos ágeis. Além disso, práticas de segurança incorporadas no ciclo de vida de desenvolvimento de software diminuem a probabilidade de existirem vulnerabilidades no sistema (HOWARD; LIPNER, 2006).

É importante para profissionais de equipes ágeis que têm interesse em implantar segurança em seus projetos, realizar um estudo que identifique, descreva e avalie o estado da prática na utilização de técnicas e ferramentas para detecção de vulnerabilidades por membros de equipes que utilizam métodos ágeis como processo de desenvolvimento de software.

1.1 Motivação

A detecção de vulnerabilidades é uma atividade essencial nos processos de desenvolvimento de software. Por isso é importante utilizar um processo de desenvolvimento seguro, pois pode diminuir a probabilidade de ocorrerem vulnerabilidades no software, já que um conjunto de técnicas de segurança serão aplicados em cada fase do ciclo de vida do desenvolvimento. Atividades de segurança não devem ser aplicadas apenas nas fases finais do desenvolvimento de software, mas sim aplicadas de forma contínua desde as fases iniciais.

Pelo fato de os métodos ágeis trabalharem com incrementos rápidos, é importante avaliar as práticas de segurança aplicadas pelos membros das equipes e se estes vêm obtendo êxito.

1.2 Justificativa

É importante adotar práticas de segurança, sejam elas processos, atividades, técnicas e ferramentas, sejam elas modelos para melhorar a segurança de suas aplicações. Isso é motivado pelos ataques cibernéticos cada vez mais frequentes, o que exige das organizações políticas de segurança efetivas (VIANNA; FERNANDES, 2015). Elas têm, porém, dificuldades em controlar as vulnerabilidades pois são muitas as existentes (OWASP, 2010b). De acordo com a OWASP (2010b), alguns exemplos de vulnerabilidades conhecidas são: injeções de falha, *cross site scripting*, exposição de dados sensíveis, entre outras.

Os métodos ágeis têm se tornado prevalentes na indústria de software por utilizar práticas baseadas em pouca documentação, interação com o cliente, respostas a mudanças, muita comunicação verbal e foco na geração de código executável da aplicação (BECK; BEEDLE et al., 2001).

Por meio de uma revisão bibliográfica da literatura foram encontrados diversos processos de desenvolvimento seguro de software como: Processo de McGraw (McGraw's process), Ciclo de Vida de Desenvolvimento de Segurança da Microsoft (*Microsoft Security Development Life Cycle*), Modelo de Desenvolvimento Seguro de Software (*Secure Software Development Model*), Processo Leve e Abrangente de Aplicação de Segurança (*Comprehensive, Lightweight Application Security Process*) e Processo para Equipes de Desenvolvimento de Software Seguro (*Team Software Process for Secure Software Develop-*

ment). Destes processos identificados, nenhum está voltado exclusivamente para métodos ágeis. Sendo assim, Howard e Lipner (2006) incentivam o uso de algumas atividades, técnicas e ferramentas voltadas para o contexto de desenvolvimento ágil de software.

Portanto, faz-se necessário avaliar como os membros das equipes, que utilizam métodos ágeis, estão incorporando técnicas e ferramentas para detecção de vulnerabilidades em seus processos de desenvolvimento de software. Além disso, é preciso identificar quais as motivações, quais os benefícios e as lições aprendidas na adoção dessas técnicas e ferramentas.

A partir dessa avaliação, foi possível identificar as dificuldades e oportunidades para conciliar métodos ágeis e segurança de software.

1.3 Objetivos

O objetivo geral deste estudo consiste em identificar, descrever e avaliar o estado da prática na utilização de técnicas e ferramentas para detecção de vulnerabilidades, por membros de equipes que utilizam métodos ágeis como processo de desenvolvimento de software.

Para atingir o objetivo principal, os seguintes objetivos específicos foram definidos:

- Descrever as características dos membros das equipes de desenvolvimento ágil de software.
- Descrever as características dos projetos que utilizam métodos ágeis e segurança de software.
- Identificar as técnicas e ferramentas para detecção de vulnerabilidades utilizadas pelos membros das equipes de desenvolvimento ágil de software.
- Identificar o grau de implantação e benefícios na implantação de técnicas e ferramentas para detecção de vulnerabilidades.
- Identificar as necessidades de treinamento dos membros de equipes em termos de técnicas e ferramentas para detecção de vulnerabilidades em desenvolvimento ágil de software.
- Identificar as motivações, as estratégias de implantação, os porquês, as dificuldades, as limitações e as lições aprendidas durante o uso das técnicas e ferramentas.

1.4 Resultados obtidos

Foram obtidas por meio de estatística descritiva as características dos membros das equipes como função atual e anos de experiência com desenvolvimento ágil de software.

Já em relação às características dos projetos, foram descritos quais os métodos ágeis utilizados pelas equipes, as metodologias utilizadas para a implantação de segurança, o número de membros da equipe de segurança e as tecnologias utilizadas. Em relação ao estado da prática, foram identificadas as técnicas e ferramentas utilizadas na detecção de vulnerabilidades em cada fase do ciclo de desenvolvimento ágil de software, o grau de aptidão dos membros nas técnicas e ferramentas e as necessidades de treinamento. Foi possível identificar o grau de implantação das técnicas e ferramentas e os benefícios observados pelos gerentes de equipes ágeis.

As características que influenciam na implantação como as motivações, os porquês, as estratégias adotadas, as dificuldades, as limitações e as lições aprendidas foram identificadas e suas respostas interpretadas. Finalmente, foram identificadas e descritas as possíveis influências que acarretam na implantação ou não das técnicas e ferramentas no desenvolvimento ágil.

1.5 Estrutura da dissertação

Este capítulo apresentou a motivação, a justificativa, o objetivo e os resultados obtidos de forma resumida.

O restante deste trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta o conceito de métodos ágeis, seus valores e princípios, principais métodos ágeis utilizados na indústria.
- O Capítulo 3 apresenta o conceito de vulnerabilidade, as técnicas e ferramentas para detecção de vulnerabilidades e as ferramentas que apóiam essas técnicas.
- O Capítulo 4 apresenta os processos existentes para segurança em desenvolvimento de software, com ênfase em recursos de segurança.
- O Capítulo 5 discute os trabalhos relacionados.
- O Capítulo 6 apresenta os procedimentos metodológicos utilizados na pesquisa e a abordagem do *survey*.

- O Capítulo 7 apresenta a análise dos dados.
- Finalmente, no Capítulo 8 são apresentadas as conclusões e os trabalhos futuros.

2 Métodos ágeis

Este capítulo inicialmente apresenta a história dos métodos ágeis. Além disso, são discutidos os conceitos sobre métodos ágeis, seus valores e princípios, e detalhados os principais métodos ágeis, suas práticas, atividades e processos.

2.1 História

De acordo com Sommerville (2011) no final da década de 1980 e início da década 1990 a opinião predominante na Engenharia de Software era de que o melhor jeito para alcançar o sucesso de um projeto de software seria por meio de um planejamento criterioso do projeto de software, com a utilização de documentações bem detalhadas, métodos de análise e projeto apoiado por ferramentas CASE (*Computer-Aided Software Engineering*) e processos de desenvolvimento de software controlados e rigorosos. Tal visão originava-se da comunidade de Engenharia de Software responsável pelo desenvolvimento de grandes sistemas de software, como sistemas aeroespaciais e de governo (PRESSMAN, 2009). Nesses tipos de sistemas, as equipes devem seguir um plano pré-estabelecido de desenvolvimento de software e estruturado de forma sequencial, no qual as atividades posteriores só podem ser iniciadas quando as anteriores forem finalizadas (SOMMERVILLE, 2011).

O problema dos modelos tradicionais de desenvolvimento é que equipes trabalham no projeto por um longo período de tempo e, nesse ínterim, as condições de mercado mudam rapidamente, as necessidades dos usuários não são mais as mesmas e novas ameaças competitivas surgem (PRESSMAN, 2009).

A insatisfação com estas abordagens pesadas, rígidas e criteriosas levou um grupo de especialistas em desenvolvimento de software no final da década de 1990 e início do ano 2000 a propor novos métodos para desenvolvimento de software, conhecidos como métodos ágeis (BECK; BEEDLE et al., 2001). Um método ágil, por princípio, deve estar com uma equipe rápida e capaz de responder às mudanças. Mudanças estão relacionadas ao mercado, aos requisitos, aos membros da equipe, às tecnologias e qualquer outra coisa que poderá impactar no produto em construção ou no projeto que cria o produto (BECK; BEEDLE et al., 2001).

Uma equipe ágil reconhece que o software é desenvolvido por indivíduos trabalhando em equipes e que as habilidades dessas pessoas estão no cerne do sucesso do projeto (JACOBSON, 2016). Métodos ágeis contam com uma abordagem iterativa e incremental para a especificação de requisitos de software, desenvolvimento e entrega (SOMMERVILLE, 2011).

Os engenheiros de software e outros envolvidos no projeto trabalham conjuntamente em uma equipe ágil, a qual possui capacidade de auto-organização, rápida comunicação e fácil colaboração entre os membros da equipe, o que é muito relevante para o sucesso de um projeto (BECK; BEEDLE et al., 2001).

Assim é importante existir uma alternativa a Engenharia de Software convencional, voltada para a entrega rápida de sistemas corretos e com qualidade. Métodos ágeis desenvolvem-se para sanar fraquezas reais e perceptíveis da Engenharia de Software convencional (SOMMERVILLE, 2011).

2.2 Valores e princípios dos métodos ágeis

Na visão de Pressman (2009) a agilidade em um projeto incentiva a estruturação e atitudes em equipe, o que consequentemente torna a comunicação entre membros da equipe e *stakeholders* mais fácil. Assume-se também que o cliente faz parte da equipe, além de reconhecer que o planejamento do projeto deverá ser flexível e um software funcionando corretamente deverá ser entregue rapidamente. Sendo assim, Beck, Beedle et al. (2001) definiram no Manifesto Ágil os valores e princípios dos métodos ágeis. Dentre os valores, os seguintes foram definidos:

- **Indivíduos e interações acima de processos e ferramentas:** os softwares são construídos por uma equipe e não apenas um indivíduo. Os membros da equipe precisam trabalhar juntos, caso contrário, as ferramentas e processos não terão utilidade alguma.
- **Software operacional acima de documentação completa:** o objetivo principal do desenvolvimento ágil de software é criar software. A documentação deve existir para auxiliar os *stakeholders* a entenderem como o sistema foi construído, contemplando nela apenas o necessário.

- **Colaboração dos clientes acima de negociação contratual:** alguns clientes não sabem explicar de forma clara os requisitos desejáveis em um software. Assim, mudanças frequentes nos requisitos podem existir. Ter um contrato é importante para definir as responsabilidades, custos e especificações, mas sem substituir a comunicação.
- **Respostas a mudanças antes de seguir um plano:** mudança é algo comum em um projeto. A equipe deve ter a capacidade de se adaptar às possíveis mudanças e o software precisa ser flexível o suficiente para comportá-las.

O Manifesto Ágil não rejeita processos e ferramentas, documentação, negociação de contratos ou o planejamento, mas mostra que eles têm importância secundária quando comparado aos indivíduos e interações, software funcionando, colaboração com o cliente e as respostas rápidas a mudanças e alterações (BECK; BEEDLE et al., 2001). Ainda foram definidos por Beck, Beedle et al. (2001) no Manifesto Ágil, 12 princípios que fundamentam os métodos ágeis que são:

- **Satisfação do cliente por meio da entrega frequente e contínua de software:** o objetivo do desenvolvimento de software deve ser a produção do software dentro do especificado e com entregas frequentes.
- **Aceitar mudanças de requisitos mesmo em fases avançadas do projeto:** os requisitos mudam durante o projeto. Assim, é importante que o projeto tenha flexibilidade para mudanças, inclusive em fases mais avançadas do projeto.
- **Entregar software frequentemente e na menor escala de tempo possível:** entregar frequentemente software a fim de obter *feedback* constante, mostrando o estado atual do projeto e oferecendo oportunidades para as partes interessadas direcionarem a equipe de desenvolvimento.
- **A equipe de desenvolvimento e cliente devem trabalhar juntas diariamente:** é importante a participação ativa dos *stakeholders* no projeto, com reuniões, adoção de ferramentas e técnicas que permitam às partes interessadas serem envolvidas de forma ativa no projeto.
- **Equipe com indivíduos motivados, oferecendo a eles um ambiente de confiança, segurança e apoio:** perceber a importância de ter equipes motivadas, qualificadas, apoiadas por seus gestores e colaborando uns com os outros é um dos fatores de sucesso de um projeto.

- **A melhor forma de transmitir informações para a equipe é por meio de conversas presenciais:** é necessário ter um canal de comunicação eficaz e efetivo entre membros da equipe e os *stakeholders*. Dentre as diversas formas de comunicação, a mais utilizada e recomendável é a conversa presencial.
- **Software funcional é a medida primária de progresso:** a principal medida de progresso do desenvolvimento de software deve ser a entrega de funcionalidades de software que atenda às necessidades de seus *stakeholders*.
- **Processos ágeis promovem um ambiente sustentável:** é importante manter um ritmo constante no projeto. Não alcançará a excelência um projeto em que a equipe se dedica muitas horas e realizando horas-extra.
- **Atenção à excelência técnica e bom *design* aumenta a agilidade:** é necessário seguir diretrizes de desenvolvimento, em especial, diretrizes de codificação, boas práticas e orientações de modelagem para alcançar a qualidade e agilidade.
- **Simplicidade é essencial em projetos com métodos ágeis:** desenvolver apenas o necessário, a fim de garantir a funcionalidade solicitada. Melhorias virão nas próximas iterações.
- **Os melhores requisitos, arquiteturas e *design* surgem de equipes organizadas:** a equipe deve ser organizada para conseguir responder a mudanças. Os melhores resultados emergem de equipes auto-organizadas.
- **Efetividade e otimização constantes:** em intervalos regulares, a equipe deve refletir sobre como se tornar mais efetivo, realizando retrospectivas e melhorias contínuas dos processos do ciclo de vida de desenvolvimento software.

Nem todo método ágil aplica estes 12 princípios. Cada um deles adapta estes valores e princípios de diferentes formas, mas sempre mantendo a essência ágil.

2.3 Principais métodos ágeis

Cada um dos métodos ágeis possui diferentes ideias, processos e atividades. Porém, todos estes compartilham de forma geral a essência, os princípios e valores do manifesto ágil e por isso têm muito em comum (SOMMERVILLE, 2011). Exemplos de métodos ágeis são: Programação Extrema (XP — *Extreme Programming*) (BECK, 2000), Scrum (SCHWABER; BEEDLE, 2002), Lean (POPPENDIECK; POPPENDIECK, 2003), Kanban

(SKARIN, 2015), Método Cristal (*Crystal Method*) (COCKBURN, 2004), Método de desenvolvimento de sistemas dinâmicos (DSDM — *Dynamic Systems Development Method*) (STAPLETON, 1999), Desenvolvimento Guiado por Funcionalidades (FDD — *Feature-Driven Development*) (PALMER; FELSING, 2002) e Processo Unificado Ágil (AUP — *Agile Unified Process*) (AMBLER, 2002). O sucesso destes métodos resultou na noção de modelagem ágil (AMBLER, 2006) e instâncias ágeis do Processo Unificado da Rational (RUP — *Rational Unified Process*) (LARMAN, 2012).

Para exemplificar a aplicação dos conceitos de métodos ágeis serão descritos dois métodos ágeis muito utilizados: Programação Extrema (XP) e Scrum.

2.3.1 Programação extrema - XP

Programação Extrema - XP é um método de desenvolvimento ágil de software focado na excelência de suas aplicações, a qual aplica o uso de boas práticas de programação, clareza na comunicação entre seus membros e trabalho em equipe (BECK, 2000). Assim, Beck (2000) lista os seguintes aspectos de XP:

- Filosofia de desenvolvimento de software baseada nos valores de comunicação, *feedback*, simplicidade, coragem e respeito.
- Um conjunto de práticas para desenvolvimento de software que complementam uma a outra, ampliando assim os seus efeitos.
- Um conjunto de princípios e técnicas que traduzem valores em prática.
- Uma comunidade que compartilha esses valores e as mesmas práticas.

Valores

Beck (2000) define um conjunto de cinco valores que devem ser seguidos como um guia para as atividades, ações e tarefas específicas da XP. São eles:

- **Comunicação:** para conseguir a comunicação efetiva entre os envolvidos no projeto, é necessário a colaboração entre clientes e equipe. A comunicação no projeto é necessária para manter o senso de equipe e cooperação entre seus membros.

- **Simplicidade:** é um dos valores mais intensos da XP. Dessa forma, para alcançar a simplicidade, é restringido aos desenvolvedores a projetarem apenas as necessidades imediatas.
- **Feedback:** o *feedback* provém de várias fontes:
 - Opiniões vindas da equipe ou *stakeholders* sobre uma ideia proposta.
 - Da implementação da ideia.
 - Da escrita dos testes.
 - Da execução dos testes.
 - Do funcionamento após implantação.
- **Coragem:** uma equipe XP deve ter disciplina (coragem) para projetar hoje reconhecendo que as necessidades futuras podem mudar drasticamente.
- **Respeito:** na equipe XP deve haver respeito entre seus membros e entre outros envolvidos.

Práticas e conceitos

As práticas que compõem a XP melhoram o processo de desenvolvimento de software, sendo necessário decidir a ordem de aplicação de tais práticas de acordo com o ambiente do projeto e suas adaptações (BECK, 2000).

Beck (2000) lista as principais práticas que compõem a XP, que são:

- **Sentar-se juntos:** o desenvolvimento de software deve ter espaço para todos os envolvidos no projeto proporem ideias. Equipe e cliente devem manter a proximidade física, a fim de melhorar a comunicação durante o projeto e sua produtividade.
- **Equipe completa:** incluir na equipe pessoas com diferentes habilidades e perspectivas é necessário para o projeto ter sucesso. É a disponibilização de diferentes recursos no projeto. Ter senso de equipe entre os membros, interações entre membros, cooperação e compartilhamento de conhecimento são uma das práticas importantes da XP.
- **Ambiente de trabalho informativo:** um observador deve ser capaz de entrar no ambiente de trabalho da equipe e obter facilmente informações sobre o andamento do projeto, como será realizado e possíveis problemas e soluções. Todos os envolvidos no projeto devem ter acesso fácil às informações.

- **Trabalho energizado:** trabalhar por muitas horas em um projeto faz com que a equipe seja produtiva, porém torna-se difícil manter o ritmo durante os outros dias, prejudicando assim o desenvolvedor e a equipe. Os membros devem estar com mente relaxada, concentrada e preparada para trabalhar.
- **Programação em pares:** pares de programadores trabalham em uma tarefa juntos, proporcionando assim o diálogo. Enquanto um desenvolvedor verifica possíveis defeitos, ou sugere ideias e soluções para algum problema, o outro codifica. Frequentemente são revezadas as funções de cada um, bem como o par de programadores.
- **Histórias de usuário:** utiliza cartões que descrevem as unidades de funcionalidade do sistema e estimam o esforço necessário para desenvolver uma funcionalidade. Essa estimativa é realizada no início do projeto. Uma história de usuário é composta por três frases escritas pelo cliente, de forma simples e que abranja as suas necessidades.
- **Ciclo semanal:** o trabalho é planejado durante a reunião semanal. Nesta, são realizadas revisões do calendário, retrospectiva do projeto, acompanhamento do progresso do projeto, estado atual e análise das expectativas.
- **Integração contínua:** a equipe divide, conquista e integra o problema. Tem por objetivo integrar continuamente o código produzido e testado, a fim de verificar se alterações ou novas funcionalidades incorporadas resultam em possíveis defeitos. É recomendável que seja realizada frequentemente, de forma sincronizada, simples e independente.
- **Test-first programming:** consiste na escrita de testes falhos antes da escrita de código. Criar testes falhos antes da codificação auxilia o desenvolvedor a trabalhar apenas o necessário. Um teste é escrito para analisar um problema em questão, em seguida seu código é desenvolvido para passar no teste.
- **Design incremental:** investe no projeto do sistema todos os dias, e de forma iterativa e incremental a aplicação surge. O *design* é iniciado considerando os aspectos mais simples para o melhor elaborado. *Design* incremental é refinado com a experiência adquirida a cada dia por seus membros.

Processos

De acordo com Pressman (2009) a XP emprega uma abordagem orientada a objetos que envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas:

- **Planejamento:** é realizado um levantamento de requisitos utilizando histórias de usuários para entender o ambiente de negócios do software. A velocidade do projeto está relacionada à quantidade de histórias implementadas nas iterações, assim, é realizada uma estimativa para as próximas entregas.
- **Projeto:** o projeto XP segue rigorosamente o princípio de simplicidade. Os artefatos de projeto da XP são os cartões de classe-responsabilidade colaboração (CRC — *Class-Responsibility-Colaborator*), criados para serem um mecanismo eficaz para pensar sobre o software.
- **Codificação:** após desenvolver as histórias de usuário e os artefatos, a equipe desenvolve diversos testes de unidade. A XP recomenda que a codificação seja realizada em pares para auxiliar na resolução de problemas e garantia da qualidade.
- **Testes:** já foi observado que a criação de testes de unidade antes de iniciar a codificação é um elemento-chave da abordagem XP. Os testes de integração e validação do sistema podem ser realizados diariamente. Os testes de aceitação são obtidos de histórias de usuários implementadas.

2.3.2 Scrum

Scrum é um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produzem e entregam produtos com o mais alto valor possível. Cada um de seus componentes tem um propósito específico o qual é essencial para obter sucesso (SCHWABER, 2013).

De acordo com Schwaber (2013) Scrum possui os seguintes papéis-chave:

- **Scrum master:** responsável por garantir que o Scrum seja entendido e aplicado pela equipe.
- **Proprietário do produto:** é o responsável por maximizar o valor do produto e do trabalho do time de desenvolvimento.

- **Time de desenvolvimento:** são profissionais que realizam o desenvolvimento e entrega de uma versão funcional do sistema.

Valores e Fundamentos

No guia Scrum, Schwaber (2013) emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos dos eventos e artefatos produzidos durante o projeto. Assim, Schwaber (2013) lista os três pilares do Scrum que apoiam a implementação de um controle de processo que são:

- **Transparência:** aspectos significativos do processo devem ser visíveis, a fim de obter-se um padrão comum e que os observadores compartilhem um mesmo entendimento sobre o que está sendo visto.
- **Inspeção:** os usuários Scrum devem frequentemente inspecionar os artefatos do projeto, seu progresso e variações.
- **Adaptação:** se algum aspecto do processo tiver um desvio, o processo ou o artefato que está sendo produzido deverá ser ajustado.

Eventos e Artefatos Scrum

Schwaber (2013) lista os eventos e artefatos do Scrum, utilizados para criar uma rotina e minimizar a necessidade de reuniões não planejadas, são eles:

- ***Sprint*:** é uma iteração que tem por objetivo a entrega de um incremento funcional de software. Um *Sprint* tem início após a conclusão do *Sprint* anterior e tem duração entre duas a quatro semanas.
- **Reunião de planejamento do *Sprint*:** define o trabalho a ser realizado no *Sprint*. O time de desenvolvimento trabalha para estabelecer as funcionalidades que serão desenvolvidas durante o *Sprint* corrente e como irá construí-las.
- **Reunião diária:** a reunião diária do Scrum é um evento limitado a 15 minutos, para sincronizar as atividades e criar um plano com as atividades realizadas, problemas e planos para o próximo *Sprint*.
- **Revisão do *Sprint*:** o Time Scrum e as partes interessadas colaboram sobre o que foi feito e decidem as próximas atividades.

- **Retrospectiva do *Sprint*:** é uma oportunidade para o Time Scrum realizar inspeções e organizar um plano para possíveis melhorias para os próximos *Sprints*.
- **Artefatos do Scrum:** representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação.
- ***Backlog* do produto:** é uma lista ordenada das necessidades do produto, como funções, requisitos, melhorias e correções que devem ser feitas nas futuras versões do produto.
- ***Backlog* do *Sprint*:** é a definição sobre qual funcionalidade estará na próxima iteração e qual o esforço necessário para entregar essa funcionalidade.
- **Incremento:** é a soma de todos os itens do *Backlog* do produto completados durante o *Sprint*.

2.4 Aplicação dos métodos ágeis

Métodos ágeis foram desenvolvidos para serem utilizados por pequenas equipes de programação e que atuem junto no desenvolvimento de pequenos e médios sistemas (SOMMERVILLE, 2011). Para Pressman (2009) o desenvolvimento ágil oferece benefícios importantes, no entanto, não é indicado para todos os tipos de projetos, softwares e situações. Para utilizá-lo, é essencial que a equipe adapte os processos e atividades existentes e mantenha apenas artefatos realmente essenciais, utilizando processos enxutos, enfatizando a estratégia de entrega incremental.

De acordo com Leffingwell (2007) em sistemas críticos, que utilizam um processo de desenvolvimento de software rígido, é importante adaptar suas atividades para manter a essência dos métodos ágeis que é a flexibilidade, entregas frequentes de versões do sistema, integração contínua, o desenvolvimento orientado a testes e comunicação frequente entre os *stakeholders*.

Assim, Leffingwell (2007) lista uma das possíveis adaptações das atividades dos processos tradicionais de desenvolvimento de software em métodos ágeis com incorporação de práticas de segurança:

- No desenvolvimento de sistemas de grande porte não é possível focar apenas no código, então será necessário realizar atividades de projeto de arquitetura de software e documentações que descrevam os aspectos mais críticos do sistema.

- Mecanismos de comunicação eficientes e eficazes entre as equipes deverão ser desenvolvidos e utilizados. Deverá ser introduzida uma variedade de canais de comunicação, como *e-mail*, ferramentas de videoconferências e mensagens instantâneas para facilitar a comunicação.
- É essencial manter o sistema atualizado, realizando entregas regulares de versões atualizadas do sistema. Além disso, novas ferramentas de gerenciamento de desenvolvimento de software deverão ser introduzidas.

Grandes empresas têm adotado alguns princípios e fundamentos dos métodos ágeis em projetos específicos. Porém, Lindvall et al. (2004) justifica que realizar a adaptação dos métodos ágeis aos processos de desenvolvimento de sistemas críticos pode ser difícil pelas seguintes razões:

- Gerentes de projeto que não têm experiência em projetos com métodos ágeis podem não aceitar a aplicação de uma nova abordagem, pelo risco que este pode ocasionar nos projetos.
- Empresas de grande porte possuem procedimentos de qualidade e padrões que são aplicados em todos os projetos e devido à natureza burocrática dos seus processos, introduzir métodos ágeis pode ser incompatível.
- Métodos ágeis funcionam melhor quando os membros da equipe possuem habilidade em realizar a análise de requisitos corretamente, em desenvolver e testar corretamente um software, em manter uma boa comunicação com os demais membros da equipe e em adaptar-se rapidamente a possíveis mudanças no projeto. No entanto, dentro de grandes organizações existem pessoas com diferentes habilidades e conhecimentos. Assim, adotar um método ágil pode fazer com que pessoas com menos conhecimento e habilidade sejam excluídas da realização de alguma atividade.
- Pode haver resistência em utilizar métodos ágeis, especialmente nas organizações em que são utilizados os processos convencionais da Engenharia de Software.

Gestão de mudanças e procedimentos de testes são exemplos de atividades que podem não ser compatíveis com métodos ágeis. Gestão da mudança é o processo que controla alterações em um sistema, de modo que o impacto da alteração e custos sejam controlados (PRESSMAN, 2009).

Em organizações de grande porte, procedimentos de testes entram em conflito com as abordagens utilizadas nos métodos ágeis, já que nestes os procedimentos de teste devem

ser realizados a todo momento, inclusive antes do desenvolvimento (SOMMERVILLE, 2011).

De acordo com Howard e Lipner (2006) a adoção de métodos ágeis em organizações de grande porte é um processo de mudança cultural que leva muito tempo a ser consolidada, requerendo uma mudança na forma de gestão antes de ser aplicada.

Assim, organizações de grande porte que desejam usar métodos ágeis no desenvolvimento de software, precisam dedicar tempo e recursos significativos para o processo de mudança (SOMMERVILLE, 2011).

2.5 Considerações finais

Este capítulo apresentou o conceito de métodos ágeis, suas principais características, práticas, valores e fundamentos. O próximo capítulo irá apresentar os conceitos de vulnerabilidade, uma listagem das principais vulnerabilidades de software, e as principais técnicas e ferramentas para detecção de vulnerabilidades de software.

3 Vulnerabilidades

Uma vulnerabilidade em software é um conjunto de condições que podem levar à violação de uma política de segurança (SEACORD; HOUSEHOLDER, 2005). 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, 2010b). Em um sistema, procedimentos de segurança, controles internos, ou de implementação que podem ser explorados por ameaças são considerados vulnerabilidades (NIST, 2009). 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.

Quando é mencionado o termo *vulnerabilidade* em software, vários outros correlatos vêm juntamente a este, como é o caso de *ameaças* (GRIESKAMP et al., 2002). Uma ameaça é uma ocorrência que pode acarretar dano para o sistema ou organização em um ativo ou grupos de ativos. (STALLINGS, 2012). 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 a premeditação, como um monitoramento não autorizado por *hackers* (PINHEIRO, 2011). 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 (CERT.BR, 2012).

Um ataque ocorre quando uma ameaça intencional é realizada por motivos variados, como: ganhos financeiros, venda de informações, espionagem ou sabotagem (PINHEIRO, 2011). Uma das possíveis causas de ataque é a dificuldade em desenvolver softwares com a devida segurança (VIANNA; FERNANDES, 2015). Outro termo associado a vulnerabilidades é o *risco*, o qual pode ser definido como a probabilidade da ocorrência de uma ameaça (VERDON; MCGRAW, 2004). Uma vulnerabilidade também está ligada ao termo *falha* que pode ser definido como a incapacidade de executar uma função requerida dentro dos limites especificados (GROUP, 2010). De acordo com Weber (2003),

componentes físicos envelhecem e sofrem com interferências externas, seja ambiental, seja humana. Assim, as falhas geralmente são classificadas em falhas físicas, aquelas de que padecem os componentes, e falhas 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 (WEBER, 2003).

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 (MCGRAW; MORRISETT, 2000). Uma vulnerabilidade também está ligada ao termo *engano* que é definido como uma ação humana que produz um defeito (GROUP, 2010). 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 (JONES, 2013).

3.1 Principais vulnerabilidades de software

Fundada em 2 de dezembro de 2001, a *Open Web Application Security Project* - OWASP é uma comunidade aberta dedicada a capacitar as organizações a desenvolver, adquirir e manter aplicações confiáveis (OWASP, 2010b). Tal comunidade é uma das principais referências sobre segurança no desenvolvimento de aplicações web. Segundo OWASP (2010b), aplicações seguras são resultado da decisão de produzi-las por parte da organização.

Em 2010 foi desenvolvido pela OWASP (2010b) um material conhecido por *Top Ten OWASP*, que tem por objetivo fornecer informações de segurança necessárias para o desenvolvimento de aplicações web, além de educar desenvolvedores, *designers*, arquitetos e organizações a respeito das consequências das vulnerabilidades mais comuns encontradas em aplicações web.

Assim, OWASP (2010b) lista as principais vulnerabilidades em aplicações web:

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

- ***Cross site scripting - XSS:*** ocorre em aplicações que passam informações do usuário ao navegador sem validar ou codificar seu conteúdo. Para evitar essa vulnerabilidade, é necessário verificar se todos os parâmetros da aplicação são validados ou codificados antes de serem incluídos em páginas HTML (*HyperText Markup Language*).
- **Referência insegura direta a objetos:** ocorre quando um desenvolvedor expõe uma referência a um objeto de implementação interna como um registro na base de dados ou um parâmetro de um formulário. Um atacante pode manipular referências a objetos para acessar outros sem autorização.
- ***Cross site request forgery - CSRF:*** força o navegador da vítima a enviar requisições para uma aplicação web vulnerável, realizando uma ação maliciosa em nome da vítima.
- **Ausência de configurações:** é necessário definir e implantar uma configuração segura nas aplicações, *frameworks*, servidor de aplicativos, servidor *web*, servidor de banco de dados, entre outros.
- **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. Dados sensíveis merecem proteção extra, por meio de criptografia.
- **Falta de controle de nível de acesso:** as aplicações web verificam credenciais dos seus usuários em determinadas requisições e aplicativos precisam executar verificações de controle de acesso no servidor. Caso não haja um controle, será possível acessar uma funcionalidade sem autorização.
- **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.
- **Uso de componentes vulneráveis conhecidos:** bibliotecas, estruturas e outros módulos de software são executados com privilégios totais. Aplicações que utilizam componentes com vulnerabilidades conhecidas podem permitir ataques.
- **Redirecionamento e encaminhamento inválido:** aplicações web redirecionam e encaminham seus usuários para páginas utilizando dados não confiáveis para determinar as páginas de destino.

3.2 Outras vulnerabilidades

Format string: O recurso do *format string* foi desenvolvido para resolver o problema de funções de formatação de cadeias de caracteres. Dados são apresentados de acordo com uma cadeia de caracteres como entrada e avaliados como um comando (CHIEN; SZÖR, 2002). Desta forma, o atacante pode executar o código e causar uma falha de segmentação no sistema, comprometendo sua segurança.

Estouro de memória: Um estouro de memória ocorre quando uma aplicação tenta armazenar mais dados do que a memória de tamanho fixo reservado a uma variável pode acomodar. Assim, um invasor pode fazer com que um aplicativo execute ações inesperadas (SYMANTEC, 2016a). Existem ainda dois tipos de Estouro de memória

- **Stack:** utilizado em um extravasamento de memória da pilha de execução para desviar o fluxo de execução de um programa (CWE, 2015).
- **Heap:** utilizado em um extravasamento de memória para manipular o conteúdo de outra memória (CWE, 2015).

Força bruta: Um ataque de força bruta consiste em adivinhar, por tentativa e erro, as credenciais de um usuário e executar processos, acessar arquivos e serviços com os mesmos privilégios deste usuário. Tais ataques são realizados com o uso de ferramentas automatizadas (CERT.BR, 2012).

3.3 Técnicas e ferramentas para detecção de vulnerabilidades

A seguir, serão apresentadas técnicas e ferramentas para cada uma das fases do ciclo ágil de desenvolvimento de software.

3.3.1 Análise de requisitos e modelagem

A prevenção de vulnerabilidades é feita por meio da especificação de requisitos de segurança. Existem diversas linguagens para especificações de requisitos de segurança e cada uma destas possui diferentes propriedades. A seguir serão descritas algumas das linguagens de especificação utilizadas e identificadas na literatura.

Misuse cases

Misuse case é um tipo especial de diagrama de casos de uso da UML que descreve o comportamento indesejável do software (KHAN; ZULKERNINE, 2009). Linguagem de especificação de requisitos de segurança que descreve uma função que o sistema não deve permitir, como uma sequência completa de ações que resulta em perda para a organização e seus *stakeholders*. Assim, é importante empregar *misuse cases* para modelar e analisar cenários de segurança, pois auxilia na melhoria da segurança e redução de ameaças (ALEXANDER, 2003). A abordagem oferece ricas possibilidades para explorar, entender e validar os requisitos de segurança. Em OWASP (2015), é definido um *misuse case* como uma sequência de passos pelos quais um ator pode atacar um sistema. Ele pode ser usado para comunicar os potenciais riscos para as partes interessadas e detalhar e enumerar os mecanismos de defesa.

É composto por um *mis*-ator, um ator com intenção maliciosa que inicia o *misuse case*. Sindre e Opdahl (2001) apontam que, durante a modelagem, devem ser definidos primeiramente os casos de uso e atores e em seguida os *misuse cases* e *mis*-atores. As relações entre *misuse cases* e casos de uso devem ser identificadas em seguida. *Misuse cases* e *mis*-atores são representados na cor preta sólida para distingui-los dos casos de uso e atores normais. Para associar casos de uso e *misuse cases* são utilizados os relacionamentos com notação de impedimento e detecção, para indicar ações indevidas que devem ser prevenidas ou detectadas. Em seguida, novos casos de uso devem ser especificados para detectar ou prevenir ações dos *misuse cases* (SINDRE; OPDAHL, 2001).

A Figura 1 mostra um exemplo de *misuse case*, composto por atores e casos de uso representando o caso de uso, e *misuse cases* e *mis*-atores representando uma modelagem de *misuse case*.

Figura 1 – Exemplo de um *misuse case* representando um *mis*-ator realizando tentativas de ataques a um sistema de autenticação de um ator

Fonte: adaptado de Wei (2005)

Abuse cases

Abuse case é uma linguagem de especificação de requisitos de segurança baseada na interação entre atores e o sistema, no qual os resultados da interação são prejudiciais ao sistema. São definidas as interações que resultam em dano a um recurso, aos atores ou ao próprio sistema (MCDERMOTT; FOX, 1999).

O termo *abuse case* é definido como uma especificação de um tipo de interação entre um sistema e um ou mais atores, em que os resultados da interação são prejudiciais ao sistema, um dos atores, ou uma das partes interessadas (OWASP, 2010b). De acordo com McDermott e Fox (1999) os diagramas de caso de uso e *abuse case* devem ser modelados separadamente, além de não ser necessário a utilização de notação específica para modelagem. Ambos são modelados utilizando as mesmas notações comuns da UML.

Abuse cases foram adaptados a partir dos casos de uso da UML para capturar e analisar os requisitos de segurança de uma forma simples (OWASP, 2010b).

A Figura 2 mostra um diagrama de *abuse case* que foi modelado para verificar a interação entre um sistema e um ou mais atores, onde os resultados da interação são prejudiciais ao sistema.

Figura 2 – Exemplo de um *abuse case* que representa um ator malicioso realizando tentativas de ataques a um sistema de login de usuários.

Fonte: adaptado de Wei (2005)

Modelagem de ameaças

A modelagem de ameaças é uma abordagem que permite identificar, quantificar e mitigar os riscos de segurança associados a um software. A inclusão de modelagem de ameaças pode ser uma das garantias de que no desenvolvimento ágil de software estão sendo aplicadas atividades de desenvolvimento seguro desde o seu início (OWASP, 2015).

O conceito de modelagem de ameaças não é novo de modo que algumas mudanças foram surgindo. Uma das principais mudanças do conceito de modelagem foi a de olhar para um sistema com a visão de um potencial atacante, em oposição ao ponto de vista de um defensor (OWASP, 2015). De acordo com a OWASP (2015) o processo de modelagem de ameaças pode ser dividido em três etapas:

- O primeiro passo no processo de modelagem de ameaças está relacionado à compreensão da aplicação e como ela interage com entidades externas. Isso envolve a criação de *misuse cases* e *abuse cases* para identificar os possíveis pontos de entrada que um invasor pode interagir com a aplicação.
- O segundo passo é determinar e classificar as ameaças. Classificar as ameaças ajuda a identificar os pontos fracos de segurança, além de aplicar um modelo de risco baseado na probabilidade e impacto.
- O terceiro passo é determinar contramedidas e mitigação dos riscos. A falta de proteção contra uma ameaça pode indicar uma vulnerabilidade cuja exposição ao risco pode ser atenuada com a implementação de uma contramedida.

De acordo com a OWASP (2015) a modelagem de ameaças ainda é composta pelos seguintes passos:

- Compreensão da aplicação e como ela interage com entidades externas. Isso envolve a criação de casos de uso, a fim de identificar seus pontos de entrada, os potenciais invasores, os ativos e os direitos de acesso que a aplicação irá conceder a entidades externas.
- Organização de listas de classificação e categorização de ameaças, a qual tem por objetivo auxiliar a identificação de ameaças, atacantes e suas mitigações. Uma ameaça pode ser representada como uma raiz de uma árvore de ameaças e suas metas são representadas como uma árvore. *Abuse cases* representam como as atuais medidas de proteção podem ser burladas ou onde existe ausência de proteção.
- Determinar contramedidas e mitigações contra uma ameaça. Os riscos de ocorrência de ameaças são classificados e priorizados, bem como suas mitigações.

De acordo com a OWASP (2015) cada uma das etapas acima é documentada e seus resultados compõem o modelo de ameaça da aplicação. Juntamente ao documento de modelagem de ameaças, uma árvore de ameaças deve ser definida para a aplicação.

A Figura 3 mostra um exemplo de árvore de ameaças, composto por diversas ameaças e abaixo suas contramedidas.

Figura 3 – Exemplo de uma árvore de ameaças e suas mitigações

Fonte: adaptado de OWASP (2015)

3.3.2 Design

De acordo com Khan e Zulkernine (2009) a fase de *design* identifica a estrutura e os requisitos gerais do software. As linguagens de especificações de requisitos de segurança e modelagem de ameaças, por exemplo, compõem a fase de Análise de requisitos e modelagem, porém podem ser implantados na fase de *design*, abordando aspectos estruturais da segurança da aplicação e com um nível de detalhamento maior. Além disso, outras técnicas, como: *design* seguro, contramedidas de segurança e avaliação de vulnerabilidades podem ser implantadas nesta fase. A seguir serão apresentadas algumas dessas principais técnicas e ferramentas, bem como suas características e aplicabilidade.

Design seguro

As principais premissas para a fase de *design* listadas por Lipner e Howard (2005) contemplando atividades de segurança são:

- **Definir diretrizes de *design* e arquitetura de segurança:** definir a estrutura geral do software, tendo como ponto de vista a segurança e identificar os componentes cujo funcionamento correto é essencial para a segurança.
- **Identificar técnicas de *design*:** como a organização em camadas, o uso de linguagem segura, a aplicação de privilégios e a minimização da superfície de ataque.
- **Documentar elementos da superfície de ataque:** a medição dos elementos da superfície de ataque fornece à equipe métricas constantes da segurança, permitindo

a possibilidade de detecção das instâncias em que o software se torna mais suscetível a ataques.

- **Realizar a modelagem de ameaças:** a equipe realiza o processo de modelagem de ameaças, identificando as necessidades de recursos de segurança, bem como as áreas em que são necessários mais testes de segurança e revisão cuidadosa do código.

Contramedidas de segurança

Para Howard e Lipner (2006) conhecer todas abordagens usadas pelos invasores é complexo, porém, é possível listar ameaças às vulnerabilidades identificadas da aplicação. A técnica de contramedidas de segurança visa atenuar ou mitigar os riscos de possíveis ameaças às vulnerabilidades. Ela mantém uma lista de ameaças e contramedidas, agilizando o mapeamento de possíveis táticas em caso de incidentes e aplicando as contramedidas de forma mais eficaz (HOWARD; LIPNER, 2006). Essas ameaças são identificadas, categorizadas e as mitigações listadas, com o intuito de manter uma lista de ameaças e suas contramedidas. Dessa forma, as táticas listadas para as ameaças identificadas são aplicadas rapidamente em caso de incidentes.

Avaliação de vulnerabilidades

Uma avaliação de vulnerabilidade é uma auditoria interna da segurança da aplicação (MIREK et al., 2014). Do ponto de vista da fase de *design*, são identificadas as vulnerabilidades nas funções mais críticas da aplicação, mapeadas as ameaças mais conhecidas e mensurados os graus de impacto na aplicação em caso de ataques. A partir da avaliação de vulnerabilidades, são estabelecidas contramedidas de segurança com o intuito de evitar que um invasor comprometa a integridade, a disponibilidade ou a confidencialidade da aplicação (MICROSOFT, 2018a).

3.3.3 Codificação

Os artefatos produzidos nas fases anteriores fornecem uma orientação importante durante a fase de implementação. A seguir, algumas das principais técnicas e ferramentas

próprias da fase de codificação serão apresentadas, a saber: codificação segura, ferramenta de análise estática de código, revisão de segurança e ferramenta de revisão de código.

Codificação segura

A técnica de codificação segura segue padrões de segurança também definidos nas fases anteriores.

De acordo com Microsoft (2018b) a codificação segura está baseada no uso de boas práticas de codificação, ocultação de trechos de código críticos, uso de algoritmos estáveis, referência segura a objetos, reutilização de componentes seguros de software, uso de bibliotecas confiáveis e utilização de linguagem de programação segura para o problema a ser solucionado. Além disso, implementar atualizações de modo seguro, proteger as variáveis compartilhadas e os recursos contra acessos concorrentes inapropriados é uma tarefa recomendável para considerar uma codificação segura (OWASP, 2012).

Técnicas e ferramentas estáticas

Técnicas estáticas são técnicas para detecção de vulnerabilidades aplicadas diretamente no código-fonte sem executar a aplicação. De acordo com Petukhov e Kozlov (2008) exemplos de técnicas estáticas para detecção de vulnerabilidades são:

Parsing: realiza uma análise do código fonte do programa, criando uma representação do programa por meio de uma árvore sintática, a fim de analisar a sintaxe e semântica do programa (FREITEZ; MAMMAR; CAVALLI, 2009). Utilizado para examinar formatos de arquivos diferentes e analisar se o código respeita restrições sintáticas e semânticas relativas à ocorrência de vulnerabilidades, como injeção SQL e estouro de memória (SU; WASSERMANN, 2006).

Análise de fluxo de dados: refere-se a um conjunto de técnicas que derivam informações sobre o fluxo de dados ao longo da seleção de caminhos de execução de um programa (AHO; SETHI; ULLMAN, 2007).

Para Aho, Sethi e Ullman (2007) quando o comportamento de um programa é analisado, é necessário considerar todas as possíveis sequências e pontos de decisão do programa por meio de um gráfico com o fluxo que a execução do programa pode tomar.

A análise de fluxo de dados pode ser utilizada para detectar os fluxos de dados que contém vulnerabilidades (KIM et al., 2008).

Análise de fluxo de controle: é modelado como um fluxo de controle para representar blocos de código na forma de nós com dependências de controle sob a forma de arestas direcionadas (ZHIOUA; SHORT; ROUDIER, 2014). Inicialmente o fluxo é representado por um nó de entrada e concluído com o ponto final do programa. O foco principal desta técnica é determinar como são realizadas as chamadas das funções e determinar quais são realmente chamadas (ZHIOUA; SHORT; ROUDIER, 2014).

Qualificador de tipos: são utilizados para qualificar e modificar as propriedades das variáveis. São oferecidas duas propriedades a um identificador: tipo constante, que declara um objeto que não será modificado e o tipo volátil, o qual poderá ser alterado durante uma execução (MICROSOFT, 2015).

De acordo com Foster, Fähndrich e Aiken (1999) qualificadores de tipos fornecem um mecanismo para especificar e verificar propriedades de programas. Eles podem ser adicionados a qualquer linguagem como um padrão a ser seguido. Desenvolvedores adicionam qualificadores no código fonte a fim de identificar invariantes e garantir que a cada execução do programa as propriedades das anotações sejam mantidas.

Revisão de segurança de código

A revisão de segurança de código é um elemento essencial para qualquer projeto de software que tenha como objetivo a segurança (RICHARDSON; THIES, 2012).

De acordo com Richardson e Thies (2012) todo o código-fonte deve ser revisado na perspectiva de segurança. Sendo assim, os membros de equipes são encorajados a utilizarem ferramentas de revisão de segurança de código, que contribuam na busca por possíveis vulnerabilidades no código-fonte da aplicação. Além disso, durante a revisão é

analisada a compatibilidade do código-fonte com os requisitos iniciais de segurança, a fim de determinar se os objetivos foram adequadamente realizados (SYMANTEC, 2016b).

Ferramenta de revisão de código

A revisão do código é de primordial importância no desenvolvimento de software moderno (OUNI; KULA; INOUE, 2016). De acordo com Wiesmann et al. (2005) uma das principais técnicas para detecção de vulnerabilidades é a revisão de código. Elas são uma etapa essencial no processo de remoção de vulnerabilidades no software durante o processo de desenvolvimento (MICROSOFT, 2018b). A ferramenta de revisão auxilia os desenvolvedores e demais membros analisarem e avaliarem o código-fonte da aplicação antes de promovê-lo ao ambiente de produção.

Com o uso dessa ferramenta, pretende-se encontrar e corrigir erros introduzidos durante a codificação, melhorando a segurança e a qualidade do código, além de promover o aumento das habilidades dos desenvolvedores (VERACODE, 2017)

3.3.4 Testes

Técnicas e ferramentas dinâmicas

Técnicas dinâmicas são técnicas de teste de software que para detectar dinamicamente vulnerabilidades necessitam da execução do código-fonte da aplicação. Elas têm por objetivo verificar se os mecanismos de proteção construídos para o sistema irão proteger o sistema de acessos indevidos. Assim, o testador simula o papel do indivíduo que deseja acessar o sistema (SANTOS, 2011). São simulados acessos ao sistema e as tentativas de adquirir senhas por meios diversos, como: sobrecarregar, causar erros ou dominar o sistema, danificando o serviço de outros usuários (PETUKHOV; KOZLOV, 2008).

Teste de penetração

Teste de penetração é o processo de tentar obter acesso a um sistema sem conhecimento de nomes de usuários, senhas e outros meios de acesso (NORTHCUTT et al., 2006).

Seu principal objetivo é determinar como um usuário mal-intencionado pode obter acesso não autorizado aos ativos da organização (PCI, 2015).

Um teste de penetração explora as vulnerabilidades para determinar se é possível realizar o acesso não autorizado no sistema, a fim de burlar ou inutilizar os recursos de segurança (CIGITAL, 2016). Existem testes de penetração com avaliação caixa-preta no qual o cliente não fornece informações antes dos testes; avaliação caixa-branca em que a entidade fornece ao testador detalhes da aplicação; e avaliação caixa-cinza em que a entidade pode fornecer detalhes parciais dos sistemas de destino (PCI, 2015).

Fuzz testing:

Fuzz testing é uma técnica que fornece entradas inválidas, inesperadas ou aleatórias em um sistema e observa os possíveis defeitos, como lançamento de exceções imprevistas, colapso do cliente ou servidor, entre outros (FREITEZ; MAMMAR; CAVALLI, 2009). De acordo com Miller (2008) *Fuzz Testing* é mais fácil de implementar pois possui um *design* para testes mais simples e atribui entradas aleatórias para o sistema. Tal técnica possui três características:

- 1) A entrada é aleatória.
- 2) Os critérios de confiabilidade são simples, se a aplicação falha consequentemente o teste será bem-sucedido.
- 3) Como resultado das duas primeiras características, o *Fuzz testing* tem um bom desempenho e seus resultados podem ser comparados entre aplicações.

Testes baseados em riscos

O teste baseado em risco envolve testar as funcionalidades que tem maior impacto e probabilidade de falha (ISTQB, 2012). De acordo com a ISTQB (2012) eles são organizados da seguinte forma: *(i)* uma lista dos riscos priorizados é definida; *(ii)* testes que exploram cada um desses riscos é executado; *(iii)* o teste deve ser ajustado à medida que riscos são mitigados a fim de focar nos riscos atuais.

De acordo com Bach (1999) existem duas estratégias para realizar o teste baseado em riscos. A primeira é conhecida como abordagem de dentro para fora na qual é realizado

um mapeamento da aplicação, analisando possíveis problemas de segurança, seus riscos, vulnerabilidades, ameaças e as vítimas, caso incidentes ocorram.

Já a abordagem de fora para dentro possibilitam tratar os riscos de três formas:

Lista de observação de riscos: os riscos são periodicamente revisados e os testes são avaliados, a fim de verificar se eles revelam os problemas que visam solucionar.

Matriz risco-tarefa: consiste na organização dos riscos em uma tabela com duas colunas onde: são organizados os riscos de forma priorizadas e suas mitigações. Quando é analisado um risco na aplicação, a seguinte interpretação deve ser feita: há preocupação com o risco X , então é preciso investir nas tarefas Y .

Matriz componente-risco: consiste na organização dos riscos em uma tabela com três colunas. Na primeira coluna são listados os componentes da aplicação, sejam físicos ou lógicos. Na segunda suas heurísticas e na terceira suas prioridades. Essa matriz auxilia na comunicação e mapeamento de quais componentes da aplicação obterão mais esforços.

3.3.5 Produção

Gestão de respostas a incidentes

A resposta adequada aos incidentes deve ser parte integrante das políticas de segurança e da estratégia de atenuação de riscos (CICHONSKI et al., 2012). Assim, baseado no guia da *Carnegie Mellon University* proposto por West-Brown et al. (2003) um dos principais aspectos a serem especificados durante a gestão de respostas a incidentes por especialistas em segurança ou gerentes é:

- Promover a percepção da segurança dentro dos projetos.
- Estabelecer e aplicar as políticas e procedimentos de segurança.
- Avaliar regularmente as vulnerabilidades do ambiente.
- Lembrar os membros de equipes de suas responsabilidades e restrições no projeto.
- Refinar e atualizar sempre os sistemas e os procedimentos atuais.
- Manter as políticas de segurança.

Os membros de equipes devem ter a capacidade de lidar com qualquer incidente de segurança. Além disso o plano de respostas a incidentes deve ser facilmente acessível

a todos os membros de equipe, a fim de garantir que os procedimentos corretos sejam seguidos (MICROSOFT, 2018c).

3.4 Considerações finais

Este capítulo apresentou o conceito de vulnerabilidade de software e uma listagem das principais técnicas e ferramentas para detecção de vulnerabilidades em aplicações web organizadas por fase do ciclo de vida de desenvolvimento de software.

O próximo capítulo irá apresentar os principais processos de desenvolvimento de software seguro, suas atividades de segurança para cada uma das fases do ciclo de desenvolvimento e uma comparação entre os processos de segurança tradicionais e os processos voltados para métodos ágeis.

4 Processos de desenvolvimento de software seguro

Uma aplicação deve ser segura, estável e livre de riscos e danos, além de ter atividades de segurança incorporadas no seu ciclo de vida (OWASP, 2010b). Um software é considerado seguro se ele não compromete a confidencialidade, integridade e disponibilidade dos seus dados.

Tradicionalmente, a segurança do software não é tratada desde as primeiras fases do ciclo de vida de desenvolvimento de software, mas somente nas fases mais avançadas (KHAN; ZULKERNINE, 2009). Como consequência, vulnerabilidades podem ser introduzidas nas diferentes fases de desenvolvimento.

De acordo com Khan e Zulkernine (2009), a melhor maneira de desenvolver um software seguro é incorporar práticas de segurança desde o seu início. Para tanto, é necessário a utilização de processos e métodos de desenvolvimento de software seguro (*Secure Software Development*). Desta forma, um método seguro fornece maneiras de incorporar segurança no software durante as fases do ciclo de vida, por meio de: políticas de segurança, especificação de requisitos de segurança, *design* seguro, padrão de codificação segura e métodos de garantia de segurança de software como, por exemplo, testes de penetração, análise estática de código e revisões de código (HOWARD; LIPNER, 2006).

Assim, Khan e Zulkernine (2009) afirmam que existem diversos processos que incorporam atividades de segurança no desenvolvimento de software e que podem ser adaptados para a utilização em métodos ágeis. Entre os processos mencionados na literatura estão: Processo de McGraw (*McGraw's process*), Ciclo de Vida de Desenvolvimento de Segurança da Microsoft (*Microsoft Security Development Life Cycle*), Modelo de Desenvolvimento Seguro de Software (*Secure Software Development Model*), Processo Leve e Abrangente de Aplicação de Segurança (*Comprehensive, Lightweight Application Security Process*) e Processo para Equipes de Desenvolvimento de Software Seguro (*Team Software Process for Secure Software Development*).

A seguir, esses processos de desenvolvimento de software seguro são apresentados.

4.1 Processo de McGraw

O processo de segurança proposto por McGraw pode ser utilizado tanto no modelo cascata quanto no modelo incremental (POTTER; MCGRAW, 2004).

Ele abrange segurança em todo ciclo de vida de software. As atividades incorporadas são: modelagem de *abuse cases*, especificação de requisitos de segurança, análise de riscos, garantia de segurança, *misuse cases*, análise estática e dinâmica de código, teste de penetração e uso de boas práticas (KHAN; ZULKERNINE, 2009).

A Figura 4 mostra as melhores práticas de segurança de software aplicadas aos diversos artefatos do processo de McGraw.

Figura 4 – Artefatos produzidos em cada fase do ciclo de vida de desenvolvimento de software seguro do processo de McGraw

Fonte: adaptado de Potter e McGraw (2004)

4.2 Ciclo de Vida de Desenvolvimento de Segurança da Microsoft (MS SDL - Microsoft Security Development Life Cycle)

O Ciclo de Vida de Desenvolvimento de Segurança da Microsoft tem a participação do cliente na fase de análise de requisitos a fim de identificar os objetivos de segurança. O especialista em segurança atribuído ao projeto especifica critérios de saída para cada fase de desenvolvimento (KHAN; ZULKERNINE, 2009).

Para o processo MS SDL (MICROSOFT, 2016) sugerem-se as seguintes atividades para a fase de *design*: definição de uma arquitetura segura, identificação de técnicas de *design* seguras, componentes críticos para a segurança, análise de risco baseada nos componentes e modelagem de ameaças. Na fase de implementação, são utilizados padrões de codificação que são compostos por práticas como: uso de compiladores mais recentes,

organização de listas de funções proibidas ou inseguras, revisão de código e boas práticas de programação.

O processo MS SDL coloca ênfase na garantia de segurança, a qual é composta por atividades de teste de segurança e análise estática de código, que incidem especialmente sobre os componentes críticos do software (KHAN; ZULKERNINE, 2009).

4.3 *Modelo de Segurança no Desenvolvimento de Software (SSDM - Secure Software Development Model)*

O Modelo de Segurança no Desenvolvimento de Software é um processo que incorpora atividades de segurança no modelo de processo cascata (SODIYA; ONASHOGA; AJAYI, 2006).

Khan e Zulkernine (2009) citam que durante a fase de especificação de requisitos são realizadas atividades de modelagem de riscos e ameaças. A modelagem de ameaças lista as potenciais vulnerabilidades e ataques, potenciais agressores e seus objetivos. Essa lista pode ser utilizada para orientar a equipe ao longo do desenvolvimento e especialistas de segurança a definirem políticas de segurança que indiquem claramente como a segurança será alcançada. Já a análise de risco pode ser realizada em qualquer fase de desenvolvimento de software. Ela analisa as vulnerabilidades existentes, probabilidade de ataques e impactos.

Para a fase de *design*, SSDM segue as políticas de segurança definidas (SODIYA; ONASHOGA; AJAYI, 2006). Uma política de segurança é um conjunto de decisões que devem ser aplicadas durante o processo de desenvolvimento de software, inclusive do de detecção e correção de possíveis erros. Na fase de garantia de segurança são realizados testes de penetração.

4.4 *Processo Leve e Abrangente de Aplicação de Segurança (CLASP - Comprehensive, Lightweight Application Security Process)*

Processo Leve e Abrangente de Aplicação de Segurança é um processo composto por 24 atividades e recursos suplementares que tem por objetivo apoiar a construção de software seguro (GREGOIRE et al., 2007).

De acordo com OWASP (2010a), as seguintes atividades são aplicadas em cada fase do ciclo de vida de desenvolvimento de software: especificação de requisitos de segurança,

uso de *misuse cases*, análise de risco e modelagem de ameaças. Já nas fases de projeto, implementação e garantia de segurança, são modelados diagramas de classe de segurança, análise de risco, diretrizes para codificação, revisões de código, análise estática de código e testes de segurança. CLASP faz uso de uma lista composta de possíveis vulnerabilidades e as atividades de segurança que estão relacionadas a estas (OWASP, 2010a).

4.5 *Processo para Equipes de Desenvolvimento de Software Seguro (TSP Secure - Team Software Process for Secure Software Development)*

Processo para Equipes de Desenvolvimento de Software Seguro é um processo que tem por objetivo orientar as equipes no desenvolvendo de produtos de software, auxiliando as organizações a estabelecerem uma prática de engenharia madura e disciplinada, que produz rapidamente software seguro, confiável e com custos mais baixos (SEI, 2016).

As seguintes atividades de segurança compõem o processo TSP *Secure* de acordo com Khan e Zulkernine (2009).

- Planejamento da segurança.
- gerenciamento das atividades de segurança.
- Identificação dos objetivos e metas de segurança por meio da análise de riscos.
- Modelagem de *abuse cases* e modelagem de ameaças.
- Adoção de boas práticas e padrões para codificação segura.
- Atividade de revisões de código, testes unitários, *fuzz testing* e análise estática de código.

4.6 *Modelo de Maturidade para Construção de Segurança (BSIMM - Building Security in Maturity Model)*

Modelo de Maturidade para Construção de Segurança é um modelo baseado em iniciativas de segurança de software utilizadas na indústria. BSIMM não é um padrão, ao invés disso, é um conjunto de atividades de segurança de software de sucesso mundial (BSIMM, 2016).

A construção de um modelo de maturidade de segurança é um estudo das iniciativas de segurança de software existentes. Assim, ao quantificar as práticas das muitas organizações, é possível descrever o que elas compartilham em comum a cerca de segu-

rança e como aplicam tais práticas (BSIMM, 2016). É destinado para uso por qualquer pessoa responsável pela criação e execução de uma iniciativa de segurança no projeto de desenvolvimento de software (JAATUN, 2012).

Para Jaatun (2012), BSIMM e sua versão simplificada, Modelo de Maturidade para Construção de Segurança para Fornecedores vBSIMM (*Building Security In Maturity Model for vendors*), são modelos introduzidos por McGraw como tentativas de contornarem o problema de medir a segurança de software dos diversos artefatos produzidos no ciclo de vida de desenvolvimento de software, caso não seja possível medir a segurança, será possível medir seus efeitos.

O modelo BSIMM é composto por 113 atividades e 12 práticas agrupadas em quatro domínios descritos a seguir.

1) Governança: práticas que ajudam a organizar, gerenciar e medir as iniciativas de segurança de software; formação do pessoal é também uma prática de governança.

2) Inteligência: inclui um conjunto de práticas de segurança utilizadas durante o desenvolvimento de software.

3) Tópicos do ciclo de desenvolvimento de software seguro: refere-se a atividades de segurança aplicadas no ciclo de vida de desenvolvimento de software seguro, o que inclui práticas de análise de requisitos, arquitetura e *design*, implementação, revisão de código e testes de segurança.

4) Implantação: inclui práticas para teste de penetração, gerenciamento de configuração, manutenção e gestão das vulnerabilidades.

vBSIMM foi obtido a partir de atividades e práticas BSIMM que se aplicam ao desenvolvimento de software de terceiros. Fornece um método para medir a maturidade dos controles de segurança de software (ROUTH et al., 2015). A maturidade dos controles é um dos principais indicadores de potenciais vulnerabilidades de segurança em uma versão específica de software e versões posteriores pelo fornecedor (ROUTH et al., 2015).

Sendo assim, Routh et al. (2015) listam as principais fases do vBSIMM para avaliação de processos de maturidade:

- Modelagem de ameaças ou a aplicação de controles de risco na concepção de aplicações e produtos.
- Revisão de código.
- Testes de segurança.

- Teste de penetração.
- gerenciamento de configuração e resposta a incidentes.

4.7 Métodos ágeis e segurança de software

Howard e Lipner (2006) incorporam atividades de segurança aos métodos ágeis. O ponto mais importante a ser analisado é o conhecimento de políticas de segurança que a equipe possui, independentemente de qual método de desenvolvimento de software seja empregado em um projeto. Nenhum método de desenvolvimento de software cria softwares seguros se a equipe que os constrói não adotar as melhores práticas de segurança.

É necessário que a equipe seja treinada sobre questões de segurança pelo menos uma vez por ano. No custo global de desenvolvimento de software, o esforço para realização de um treinamento custa menos do que a falta de segurança (HOWARD; LIPNER, 2006).

Ao vincular métodos ágeis com recursos de segurança é necessário ter atenção, pois algumas atividades podem levar a um processo não ágil (KERAMATI; MIRIAN-HOSSEINABADI, 2008).

Métodos ágeis exigem planejamento, assim, a equipe deve decidir quem tomará as decisões de segurança no projeto. A pessoa responsável pela segurança em um projeto ágil pode ser chamada de *security advisor* ou *security coach* (HOWARD; LIPNER, 2006). Após a escolha do responsável pela segurança no projeto, é importante organizar os processos e atividades de segurança que serão aplicados no desenvolvimento ágil.

Para Keramati e Mirian-Hosseiniabadi (2008) cada método ágil possui uma estrutura composta por diferentes processos e atividades, dessa forma, tais processos e atividades devem estar adaptados para o contexto de segurança. O primeiro passo consiste em analisar os métodos ágeis e identificar seus principais processos. A partir deles, atividades de segurança devem ser definidas e englobadas em cada processo ágil.

Uma das primeiras atividades a serem realizadas é a criação de histórias de usuário, o que consiste em uma breve e simples descrição das necessidades do cliente relacionadas às ameaças percebidas por ele (HOWARD; LIPNER, 2006). O cliente é um fator-chave para a criação de histórias de usuários corretas, além de fornecer os requisitos de segurança (KERAMATI; MIRIAN-HOSSEINABADI, 2008).

Muitos softwares são complexos porque seus processos de negócios e requisitos são complexos. Assim, sempre que possível, é importante a equipe se esforçar para manter os processos e requisitos de forma simples e de fácil entendimento (HOWARD; LIPNER, 2006). A tradução das histórias de usuários em modelo de ameaças é de responsabilidade do *security coach* ou *security advisor*.

Quanto à arquitetura, é recomendável utilizar um *design* simples pois isso tem reflexos significativos na segurança do projeto. Software organizado em módulos simples tem mais chances de ser seguro, pois apenas a implementação necessária será realizada (HOWARD; LIPNER, 2006). Aplicar segurança em um projeto ágil necessita de uma modelagem de riscos e ameaças, caso contrário serão desperdiçados recursos em controles inúteis de segurança (OWASP, 2015).

Na fase de codificação, é importante aplicar boas práticas de codificação segura, revisões de segurança de código e uso de ferramentas de análise estática dentro das atividades estabelecidas para a fase. Tais atividades colaboram para manter a segurança do projeto, aplicando as definições de segurança estabelecidas pelas fases anteriores e a da própria fase (HOWARD; LIPNER, 2006).

Na fase de testes, é recomendável utilizar ferramentas dinâmicas, testes baseados em riscos ou testes que avaliem o risco e simulem ataques à vulnerabilidades, a fim de verificar o nível de segurança da aplicação. Tais técnicas e ferramentas devem ser inseridas nas atividades estabelecidas para a fase, sem perder a essência ágil (BAUMEISTER; WIRSING, 2002).

Já na fase de produção, a gestão de respostas a incidentes auxilia no gerenciamento da segurança após implantação da aplicação.

4.8 Comparação dos principais processos

O Quadro 1 apresenta uma breve comparação entre os processos de desenvolvimento de software seguro e suas atividades. A tabela está organizada da seguinte forma: F.D - Fase de desenvolvimento; A.R - Análise de Requisitos; D&P - *Design* e Projeto; Impl.- Implementação; G.S - Garantia de segurança; RC - Recursos de segurança; F.P - Uso de artefatos em fases posteriores; Ind - Uso do processo de segurança na indústria.

Quadro 1 – Comparação entre os processos de desenvolvimento de software seguro.

F.D	McGraw	MS SDL	SSDM	CLASP	TSP Secure	Howard e Lipner
A.R	Especificação de <i>abuse cases</i> , requisitos de segurança e análise de riscos.	Identificação de funções, interfaces e objetivos de segurança.	Modelagem de ameaças e definição de políticas de segurança.	Análise de riscos, modelagem de ameaças, especificação de requisitos de segurança e <i>misuse cases</i> e mitigações	Modelagem de ameaças, especificação de requisitos de segurança, <i>abuse cases</i> e análise de riscos.	Histórias de usuário, requisitos de segurança, modelagem de ameaça e risco, pequenas iterações, funcionalidades e treinamento.
D&P	Análise de riscos.	Identificação de componentes críticos, modelagem de ameaças, análise de riscos e <i>design</i> seguro.	Práticas das políticas de segurança.	<i>Design</i> seguro, uso de diagramas com notação UML, modelagem de ameaças e riscos.	Padrões de <i>design</i> seguro.	Modelagem de ameaças, <i>design</i> seguro, arquitetura simples, modelagem de riscos e técnicas de <i>design</i> seguro.
Impl.	Nenhuma.	Diretrizes de codificação segura, revisões de código e análise estática de código.	Nenhuma.	Codificação segura e boas práticas.	Codificação segura e boas práticas.	Diretrizes de codificação segura, revisões e análise estática de código, testes de segurança, ferramentas e funções seguras.
G.S	Teste baseado em risco, testes de penetração e análise estática de código.	Revisão de código e testes de segurança.	Teste de penetração.	Revisão de código, teste de penetração, análise estática e testes de segurança.	<i>Fuzz Testing</i> , teste de penetração, revisões e análise de código.	Testes de penetração, análise estática e dinâmica de código, revisões, <i>fuzz testing</i> e testes de segurança.
RC.	Diretrizes de <i>design</i> seguro e guias de segurança.	Diretrizes de <i>design</i> seguro.	Nenhum.	Diretrizes de <i>design</i> seguro, guia de boas práticas e listas de vulnerabilidades e mitigações identificadas.	Nenhum.	Plano de respostas a incidentes, guias de boas práticas e políticas de segurança.
F.P	<i>Abuse cases</i> derivam casos de testes.	Nenhum.	Nenhum.	Testes de segurança são baseados nos requisitos de segurança.	Nenhum.	Testes de segurança baseados em requisitos, revisão de segurança a cada <i>Sprint</i> .
Ind.	Relatado.	Não relatado.	Não relatado.	Relatado.	Não relatado.	Relatado.

Fonte: Lígia Santos, 2017

4.9 Considerações finais

Este capítulo apresentou os principais processos de desenvolvimento de software seguro, suas principais atividades de segurança para cada uma das fases do ciclo de desenvolvimento, utilização na indústria e uma comparação entre os processos de segurança tradicionais e processos voltados para métodos ágeis. No próximo capítulo serão apresentados os procedimentos metodológicos utilizados na condução da pesquisa.

5 Trabalhos relacionados

Este capítulo apresenta uma revisão da literatura que tem por objetivo identificar e descrever os estudos que tratam do uso de técnicas e ferramentas para detecção de vulnerabilidades (TFDV) no contexto de desenvolvimento ágil de software. Ele contém detalhes da seleção dos estudos, uma breve descrição e uma comparação entre os trabalhos selecionados e o tema desta pesquisa.

5.1 Métodos ágeis e detecção de vulnerabilidades

Esta pesquisa visa identificar e descrever o estado da prática do uso de técnicas e ferramentas de detecção de vulnerabilidades no desenvolvimento ágil de software. Para isso foi proposto um *survey* que foi aplicado para membros de equipes de desenvolvimento ágil de software que já implantaram, pretendem implantar ou estão em processo de implantação de segurança em seus processos ágeis. Dessa forma, os trabalhos identificados nesta revisão da literatura procuram relacionar atividades, técnicas e ferramentas de segurança em desenvolvimento ágil de software.

5.2 Seleção dos estudos

Além dos estudos realizados por Oyetoyan, Cruzes e Gilje (2016) e Cruzes et al. (2017), foi realizada uma revisão bibliográfica focada em estudos que tratavam de métodos ágeis e segurança no desenvolvimento de software. Para tanto, foi utilizado o repositório do Scopus¹, pelo fato de este possuir uma grande quantidade de estudos e de abranger trabalhos que também estão incluídos em outros repositórios. Assim, foi realizada uma busca por palavras-chave que tratassem sobre: *survey*, métodos ágeis, segurança de software, desenvolvimento de software seguro e engenharia de software. A cadeia de caracteres utilizada para a busca é apresentada a seguir:

¹ <https://www.scopus.com/>

Quadro 2 – Cadeia de caracteres

TITLE-ABS-KEY (secure software development AND (agile* OR method* OR vulnerab* OR secur* OR process*)) AND PUBYEAR >2010 AND (LIMIT-TO (SRCTYPE "p") OR LIMIT-TO (SRCTYPE "j")) AND (LIMIT-TO (DOCTYPE "cp") OR LIMIT-TO (DOCTYPE "ar")) AND (LIMIT-TO (SUBJAREA "COMP") OR LIMIT-TO (SUBJAREA "ENGI") OR LIMIT-TO (SUBJAREA "BUSI")) AND (LIMIT-TO (LANGUAGE "English"))

Fonte: Lígia Santos, 2018

Após a análise completa dos títulos, resumos, conclusões e referências dos estudos retornados, foram selecionados 12 trabalhos com temas próximos ao desta pesquisa.

O Quadro 3 lista os trabalhos identificados.

Quadro 3 – Trabalhos relacionados

Autores	Tema
Baca e Carlsson (2011)	Condução de entrevistas de acordo com as fases de desenvolvimento.
Baca et al. (2015)	Condução de um estudo empírico que compara processos tradicionais e ágeis seguros.
Othmane et al. (2014)	Proposta de um método de garantia de segurança em incrementos de software.
Márquez et al. (2015)	<i>Survey</i> com especialistas em segurança e arquitetos de software utilizando um método de comparação constante.
Keramati e Mirian-Hosseiniabadi (2008)	Definição de um algoritmo de tolerância à redução de agilidade para integração de atividades de segurança em métodos ágeis.
Gregoire et al. (2007)	Comparação e sugestão de novas atividades de segurança para os processos SDL e CLASP, considerando cada fase de desenvolvimento.
Bartsch (2011)	Condução de pesquisas exploratórias, qualitativas e entrevistas com especialistas em segurança que atuam no desenvolvimento ágil.
Wärynen, Bodén e Boström (2004)	Análise de XP aque apoiam o desenvolvimento de software seguro.
Beznosov e Kruchten (2004)	Classificação das atividades tradicionais de garantia de segurança e sua aderência no contexto ágil. Prós e contras das atividades.
Franqueira et al. (2011)	Proposta de incorporação de atividades de segurança nas iterações. É realizada uma avaliação de técnicas para tratamento e aceitação de riscos.
Oyetoyan, Cruzes e Gilje (2016) Cruzes et al. (2017)	Condução de <i>survey</i> para medição de habilidades de segurança de software, uso e necessidades de treinamento em equipes ágeis.

Fonte: Lígia Santos, 2018

5.3 Descrição dos trabalhos relacionados

Da revisão bibliográfica realizada, foram encontrados trabalhos que procuram identificar e avaliar o uso de segurança e métodos ágeis na indústria por meio de *surveys*, entrevistas e estudos de casos. A outra abordagem relacionada com essa pesquisa mostra

trabalhos que propõem extensões e comparações dos métodos ágeis para inclusão de atividades de segurança, bem como a comparação entre atividades de segurança dos processos tradicionais e métodos ágeis. A seguir são descritos os trabalhos obtidos, de acordo com essas duas abordagens.

5.3.1 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. Especialistas em segurança de software e arquitetos de software foram entrevistados com o intuito de encontrar teorias de segurança emergentes a partir dos artefatos de software.

O artigo permite ter uma primeira impressão do uso de artefatos de software para fazer *design* de segurança. As evidências indicam que os especialistas não concordam com a importância de analisar aspectos de segurança durante o ciclo de vida do software, nem o nível de abstração exigido para esta tarefa. Para alguns dos entrevistados, a segurança deve ser verificada em tempo de execução do projeto, tratando a segurança durante as atividades de implementação. Para outros especialistas, a segurança pode ser analisada em todo o ciclo de desenvolvimento de software. Portanto, não há uma visão comum entre os especialistas no nível de abstração necessária para analisar as preocupações de segurança.

Bartsch (2011) descrevem *insights* dos profissionais especialistas em segurança em desenvolvimento ágil a partir de pesquisas exploratórias, qualitativas e resultados de entrevistas. Foram entrevistados dez profissionais de desenvolvimento ágil a fim de melhorar a compreensão da aplicação de segurança em projetos ágeis. Os autores identificaram que os clientes e os desenvolvedores precisam de um entendimento comum de seus papéis na segurança do projeto e que esta depende da experiência dos indivíduos, seja do cliente, seja do desenvolvedor.

Baca e Carlsson (2011) apresentam um estudo realizado com desenvolvedores da Ericsson, empresa global do segmento de telecomunicações. 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. Foram conduzidas entrevistas divididas por fases: análise de requisitos, projeto, implementação, teste e implantação. A conclusão é que as atividades de segurança dos três processos não possuem uma boa aderência com processos ágeis, seja pelo alto custo, seja por não serem suficientemente benéficas para o contexto ágil. Os resultados obtidos das entrevistas mostram que as fases de *design* e teste são as mais complexas de se incorporar segurança.

Já Baca et al. (2015) descrevem um processo de desenvolvimento ágil com segurança introduzido para desenvolver um sistema de transferência de valores na Ericsson. Seu objetivo é a identificação e mitigação de riscos, enquanto mantém o custo de segurança do projeto em um nível razoável. Os dados empíricos indicam que o processo de desenvolvimento ágil com segurança supera um processo ágil tradicional, além de serem destacadas as suas restrições. Além disso, foi verificado como introduzir tarefas críticas de segurança no ciclo de vida de desenvolvimento ágil de software. No final do estudo foi realizada uma discussão sobre o custo e utilidade do método ágil proposto.

A melhoria da segurança no contexto ágil é o tema proposto nos trabalhos de Oyetoyan, Cruzes e Gilje (2016) e Oyetoyan, Jaatun e Cruzes (2017). 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. A pesquisa concluiu que: (1) as duas organizações executam diferentemente as atividades básicas de segurança de software, mas são semelhantes quanto ao aproveitamento de outras atividades, técnicas e ferramentas para segurança; (2) independentemente de custo ou benefício, habilidades direcionam o tipo de atividades que são executadas; (3) o *design* seguro é expresso como a técnica que possui a maior necessidade de treinamento por todos os grupos em ambas as organizações; e (4) a adoção efetiva de segurança de software em configuração ágil não é instantânea, requer um planejamento e direcionamento.

5.3.2 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. Para responder a esta pergunta, foi necessário questionar a possibilidade de se garantir com eficiência a segurança de incrementos de software produzidos. A resposta foi dada por meio da técnica de *caso de garantia de segurança* que apoia a segurança nos incrementos de software. O caso de garantia de segurança tem uma estrutura de árvore. A raiz é a reivindicação de segurança, os nós intermediários são os argumentos e as folhas são as evidências. Dessa maneira, atividades de engenharia de segurança são integradas em processos ágeis, reduzindo o custo de garantia da segurança do software e de mitigação de ameaças. A desvantagem dessa abordagem é que ela tem apenas a visão de segurança dos *stakeholders* não levando em consideração a visão dos atacantes.

Outra solução propõe a medição da redução da agilidade devida à inclusão de atividades de segurança em métodos ágeis. Keramati e Mirian-Hosseiniabadi (2008) definem um algoritmo de integração com um parâmetro ajustável chamado de tolerância à redução da agilidade (ART – *Agility Reduction Tolerance*). Dessa maneira, as atividades de segurança podem ser integradas a metodologias ágeis para melhorar a segurança do software. A equipe de engenharia de segurança pode ajustar o parâmetro de tolerância de redução de agilidade (ART) para um equilíbrio entre a diminuição da agilidade e as atividades de segurança.

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. Uma intersecção entre as abordagens estudadas é discutida, analisando as características dos dois processos, a intersecção de cada método e a identificação de melhorias. O processo SDL é avaliado como um processo mais pesado e rigoroso, tornando-o mais adequado para grandes organizações. CLASP, no entanto, é mais leve e acessível para pequenas organizações com demandas de segurança menos rígidas. Como conclusão, o artigo identifica as semelhanças dos processos, discute a especificidade de cada abordagem e propõe sugestões para melhorias nos processos CLASP e SDL.

Franqueira et al. (2011) analisam como as atividades de segurança em processos tradicionais de desenvolvimento de software podem ser utilizadas no contexto ágil. O trabalho classifica as atividades tradicionais, usadas para garantia de segurança, em relação a sua aceitação no desenvolvimento ágil. Também propõe maneiras de acomodar possíveis conflitos entre técnicas.

Wäyrynen, Bodén e Boström (2004) fazem uma análise do método XP a partir do modelo de maturidade da capacidade em engenharia de segurança de sistemas (SSE-CMM) ISO (2008) e *Common Criteria* (CC) CCMB (2012). Os autores mostram que o XP não possui atividades que apoiam o desenvolvimento de software seguro. Assim, as seguintes atividades deveriam ser incorporadas: (i) especificação dos requisitos de segurança; (ii) apoio a questões de segurança por meio da avaliação dos riscos de segurança; (iii) construção de argumentos de garantia; e (iv) confiança nas verificações de software. Porém, essas atividades podem demandar documentação, reduzindo assim os benefícios de XP.

O estudo de Baca e Carlsson (2011) também incorpora atividades de segurança em processos tradicionais de desenvolvimento de software seguro: Cigital *Touchpoints*, *Common Criteria* e Microsoft SDL. E Oyetoyan, Cruzes e Gilje (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.

5.4 Comparação com os trabalhos selecionados

A pesquisa descrita nesta dissertação estende o estudo experimental de Oyetoyan, Cruzes e Gilje (2016) e Oyetoyan, Jaatun e Cruzes (2017).

Primeiramente foi elaborado o convite e o instrumento do experimento, de acordo com as regras do Comitê de Ética em Pesquisa da USP (CEP), onde foi elaborado um resumo da pesquisa contendo informações sobre o tipo de pesquisa, o seu objetivo, o seu público alvo, a sua duração, o mapeamento dos possíveis riscos, contato dos pesquisadores e a garantia da confidencialidade das informações prestadas. Além disso foram especificadas também as técnicas estatísticas que seriam adotadas na pesquisa, como a técnica de análise de conteúdo que necessita da realização de gravações de entrevistas e manipulações dos dados obtidos das respostas dos questionários. Após a aprovação do convite e do instrumento pelo Comitê de Ética, foi possível dar início a pesquisa.

O *survey* foi realizado com membros de equipes ágeis selecionados por meio da rede de profissionais LinkedIn. A princípio o convite era enviado via e-mail, porém a taxa de retorno era baixa. Assim, optou-se por enviar pela rede LinkedIn havia uma maior taxa de retorno dos respondentes. Foi necessário ter essa alteração no instrumento e amostral uma vez que não foi possível identificar organizações dispostas a participar do estudo.

Assim, durante o envio dos convites, foi analisado o perfil dos respondentes, e selecionados os que tinham função ativa em equipes ágeis e que possuíam interesse em implantar, já implantaram ou estavam em processo de implantação de TFDVs em seus processos ágeis. Além disso, os participantes eram questionados se atualmente estavam atuando em alguma equipe ágil.

Semelhantemente a Oyetoyan, Cruzes e Gilje (2016) e Oyetoyan, Jaatun e Cruzes (2017), esta pesquisa avalia as aptidões, o uso atual e as necessidades de treinamento em TFDVs de membros de equipes ágeis. Além disso, ela identifica o grau de implantação, os benefícios obtidos, a motivação, as dificuldades, as limitações e as lições aprendidas no processo de implantação de TFDVs. *Insights* foram obtidos por meio da técnica de análise de conteúdo das respostas discursivas dos membros das equipes ágeis e das entrevistas com gerentes.

5.5 Considerações finais

Este capítulo apresentou uma revisão bibliográfica sobre técnicas e ferramentas de detecção de vulnerabilidades (TFDV) no desenvolvimento ágil de software. Foram encontrados os seguintes tipos de trabalhos: avaliações do uso de TFDVs em métodos ágeis, extensões dos métodos ágeis para inclusão de TFDVs e a comparação entre atividades de segurança dos processos tradicionais e métodos ágeis.

Esta pesquisa avalia as aptidões, o uso atual e as necessidades de treinamento em TFDVs de membros de equipes ágeis como em trabalhos anteriores. Ela estende esses trabalhos ao identificar os benefícios, as motivações e as lições apreendidas do uso de TFDVs no contexto brasileiro. No próximo capítulo, são apresentados os procedimentos metodológicos utilizados na condução da pesquisa.

6 Procedimentos metodológicos

Neste capítulo, é descrita a estratégia da pesquisa adotada, a relação entre objetivos, métodos e resultados desenvolvida para a pesquisa, o instrumento desenvolvido, os sujeitos participantes da pesquisa, a coleta de dados realizada e o tratamento dos dados coletados.

6.1 Estratégia de pesquisa

Como estratégia de pesquisa foi utilizado o *survey*. Segundo Isaac e Michael (1995), *survey* é um método consistente de coleta de dados que auxilia a identificação das condições de uma população ou grandes amostras, bem como as suas características, a documentação de opiniões e a comparação entre grupos. Isto é, um *survey* permite responder questões levantadas, seja para resolver problemas que foram colocados ou observados, seja para avaliar necessidades, seja para estabelecer novas metas.

Pelo fato de um *survey* possuir baixo custo e ser eficiente na coleta dos dados, ele é muito utilizado (ROBSON; MCCARTAN, 2016). No entanto, é importante notar que eles apenas fornecem estimativas para a população ou a amostra, e não medidas exatas (SALANT; DILLMAN; DON, 1994).

De acordo com Robson e McCartan (2016), *surveys* que utilizam questionários enviados por e-mail devem ter um cronograma que considere a necessidade de reenvio de convites, lembretes de participação, por exemplo, o que acaba demandando um período substancial de tempo.

Para Blair, Czaja e Blair (2013), o questionário de um *survey* deve estar alinhado aos objetivos da pesquisa. Assim, esses autores definiram um modelo que não só enfatiza a tarefa do pesquisador e entrevistador de relacionar as perguntas do questionário às questões da pesquisa, mas enfatiza a tarefa do entrevistado, em interpretar a questão, lembrar a informação relevante, julgar, decidir uma resposta e informá-la. A Figura 5 apresenta a estrutura de *design* de um *survey* que utiliza questionário como forma de coleta de dados:

Figura 5 – Modelo do processo de coleta de dados em um *survey* utilizando questionário

Fonte: adaptado de Blair, Czaja e Blair (2013)

6.2 Relação entre objetivos, métodos e resultados

Foi utilizado o instrumento denominado matriz de amarração para relacionar os objetivos da pesquisa com a fundamentação teórica, os métodos de pesquisa e os resultados esperados. Uma matriz de amarração tem por objetivo avaliar a aderência e compatibilidade entre modelo de pesquisa, objetivos da pesquisa, hipóteses de pesquisa e técnicas de análise planejadas para tratamento dos dados (TELLES, 2001).

O Quadro 4 apresenta a matriz de amarração desenvolvida para este projeto de pesquisa.

Quadro 4 – Matriz de amarração

Objetivo geral	Objetivos específicos	Fund. teórica	Pontos de investigação	Coleta de dados	Análise de dados	Resultados Esperados	Apresentação
O objetivo geral deste estudo consiste em identificar, descrever e avaliar o estado da prática na utilização de técnicas e ferramentas de detecção de vulnerabilidades por membros de equipes que utilizam métodos ágeis como processo de desenvolvimento de software.	1- Descrever as características dos membros das equipes de desenvolvimento ágil de software.	Métodos ágeis vulnerabilidades do processo de desenvolvimento de software seguro	-Qual a função dos membros que atuam em equipes de desenvolvimento ágil de software e há quantos anos atuam na atual função?	Entrevistas Questionários	Qualitativa: análise de conteúdo. Estatística descritiva	-Características como: função, na equipe, anos de experiência com desenvolvimento ágil de software.	Tabelas Quadros
	2- Descrever as características dos projetos que utilizam métodos ágeis e segurança de software.		-Quais são os métodos ágeis utilizados? Quantos membros atuaram nos projetos mais relevantes? Foi utilizada alguma metodologia para auxiliar a implantação? Qual o tipo de tecnologia utilizada nos projetos?			-Características dos projetos, como: métodos ágeis utilizados, metodologias utilizadas para implantação, números de membros na equipe e tecnologias utilizadas.	
	3- Identificar as técnicas e ferramentas de detecção de vulnerabilidades utilizadas pelos membros das equipes de desenvolvimento ágil de software.		-Quais técnicas e ferramentas são utilizadas nas fases do ciclo de vida ágil?			-Técnicas e ferramentas utilizadas em cada fase do ciclo de desenvolvimento ágil de software.	
	4 - Identificar o grau de implantação e benefícios nas técnicas e ferramentas de detecção de vulnerabilidades.		-Qual o grau de implantação e benefícios obtidos com a implantação das técnicas e ferramentas?			-Identificar o grau de implantação e benefícios obtidos com a implantação das técnicas e ferramentas no ciclo ágil.	
	5- Identificar as necessidades de treinamento dos membros de equipes em termos de técnicas e ferramentas para detecção de vulnerabilidades em desenvolvimento ágil de software.		-Qual o grau de aptidão nas técnicas e ferramentas, o grau de interesse nas técnicas e ferramentas e interesse em seus conjuntos?			-Identificar as necessidades de treinamento, os interesses e as aptidões dos membros nas técnicas e ferramentas.	
	6- Identificar as motivações, as estratégias de implantação, os porquês, as dificuldades, as limitações e as lições aprendidas.		-Qual a motivação para implantação de técnicas e ferramentas, os porquês, as estratégias, as dificuldades, as limitações e as lições aprendidas?			-Identificar as características em relação à implantação das técnicas e ferramentas, como as motivações, os porquês, as estratégias, as dificuldades e as lições aprendidas.	

Fonte: Lígia Santos, 2017

6.3 Survey

Nesta seção, serão detalhadas as práticas e padrões utilizados para o *survey* que são: descrição dos sujeitos participantes, instrumento desenvolvido para o *survey*, coleta de dados e tratamento dos dados coletados.

6.3.1 Sujeitos da pesquisa

Para a realização do *survey* e condução das entrevistas desta pesquisa, foram escolhidos membros de equipes de desenvolvimento de software e gerentes que utilizam métodos ágeis.

Foram estabelecidas cinco pessoas-chaves diretamente envolvidas na implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto: *i)* especialista em segurança; *ii)* analista de requisitos; *iii)* arquiteto de software; *iv)* desenvolvedor; *v)* testador. Alguns dos membros participantes atuam em mais de uma função, em funções diferentes das especificadas ou ainda possuem funções específicas de equipes ágeis como: *scrum master*, *product owner* e *agile coach*. Sendo assim, tais funções complementam as cinco pessoas-chaves principais da pesquisa.

Assim, foram selecionados 110 membros de equipes de desenvolvimento ágil de software por meio da busca por profissionais na rede de negócios LinkedIn¹.

6.3.2 Instrumento do survey

O instrumento do *survey* foi elaborado por meio da ferramenta *Online Pesquisa*² e é composto por um questionário de 18 perguntas aplicados aos membros de equipes ágeis e 24 perguntas para os gerentes. As suas perguntas foram baseadas na revisão bibliográfica realizada. Sendo assim, para o desenvolvimento do instrumento desta pesquisa a seguinte estratégia foi adotada:

1 - Definição de perguntas gerais sobre os membros: possibilitaram a descrição das suas características quanto a função desempenhada na equipe ágil e anos de experiência com desenvolvimento de software. Foram descritas as características dos

¹ <https://www.linkedin.com>

² <https://www.onlinepesquisa.com>

projetos, como: métodos ágeis utilizados, tamanho da equipe que implantou ou pretende implantar as técnicas e ferramentas, as tecnologias adotadas, as metodologias utilizadas e o responsável pela implantação de segurança nos projetos.

2 - Seleção de três conhecidos processos de desenvolvimento de software seguro: foram selecionados os processos: Processo de McGraw, OWASP CLASP e as atividades de Howard e Lipner pois são processos tradicionalmente conhecidos e frequentemente citados pelos autores quando o tema segurança em desenvolvimento de software é tratado.

3 - Identificação, comparação e classificação das técnicas e ferramentas: a partir da seleção dos três processos de desenvolvimento de software seguro, foram identificadas e descritas 18 técnicas e ferramentas de segurança que são comuns aos processos. Elas foram classificadas em função das fases do ciclo de vida de desenvolvimento de software: análise de requisitos e modelagem, *design*, codificação, testes e produção. O quadro 5 apresenta as técnicas e ferramentas classificadas e a figura 6 apresenta a classificação das técnicas e ferramentas nas fases do ciclo ágil.

Assim foi possível descrever o estado da prática em termos de uso atual das técnicas e ferramentas para detecção de vulnerabilidades.

Quadro 5 – Técnicas e ferramentas para detecção de vulnerabilidades utilizadas pelos processos de segurança

Análise de requisitos e modelagem	Codificação
Especialista em segurança	Codificação segura
Requisitos de segurança	Revisão de segurança de código
<i>Abuse/Misuse Cases/Stories</i>	Ferramenta de revisão de código
Modelagem de ameaças	Ferramenta de análise estática de código
Modelagem de riscos	Testes
Ferramenta de modelagem de riscos e ameaças	Ferramenta de análise dinâmica de código
<i>Design</i>	<i>Fuzz Testing</i>
<i>Design</i> de segurança	Teste de penetração
Contramedidas de segurança	Testes baseados em riscos
Avaliação de vulnerabilidades	Produção
	Gestão de respostas à incidentes

Fonte: Lígia Santos, 2017

Figura 6 – Iterações e uso das técnicas e ferramentas nas fases do ciclo de vida ágil

4 - Questões de implantação e seus benefícios: foram definidas quatro perguntas sobre o grau de implantação das técnicas e ferramentas e a ocorrência de benefícios com essa implantação. As perguntas baseiam-se nas técnicas e ferramentas adotadas dos processos de segurança selecionados, conforme detalhado no Capítulo 4.

A Tabela 29, do Apêndice A, apresenta as questões desenvolvidas no instrumento para avaliar o grau de implantação das técnicas e ferramentas, bem como os benefícios percebidos pelos gerentes. Dessa forma, cada pergunta da Tabela 29 possui um peso de 0,25 na porcentagem total de implantação de uma técnica ou ferramenta. Logo, a porcentagem de implantação é calculada da seguinte forma:

$$\%IMP \text{ Processo} = ((P1 * 0,25) + (P2 * 0,25) + (P3 * 0,25) + (P4 * 0,25))$$

onde $\%IMP$ é a porcentagem de implantação da técnica ou ferramenta e $P1$ – Pergunta 1, $P2$ – Pergunta 2, $P3$ – Pergunta 3 e $P4$ – Pergunta 4 são os valores das respostas dos participantes. As respostas variam de 0% (não implantado) até 100% (completamente implantado).

Em relação aos benefícios previstos pela literatura para determinada técnica ou ferramenta, foram definidas quatro perguntas na qual o respondente apontou “Sim” ou “Não” para cada uma destas. Dessa forma, foi possível relacionar a percepção dos benefícios obtidos com as técnicas e ferramentas e sua implantação.

5 - Questões relacionadas as aptidões e necessidades de formação em segurança: foram definidas perguntas relacionadas às aptidões e necessidades de formação dos membros de equipes ágeis. Para isso foi considerada a escala de Likert de cinco pontos, sendo: (1) - Baixa, (2) - Básica (3) - Média, (4) - Alta e (5) - Especialista. Já para o grau de interesse e necessidade em formação, a seguinte escala foi considerada, sendo: (1) - Não interessado, (2) - Levemente interessado, (3) - Moderadamente interessado, (4) - Interessado e (5) - Muito interessado.

O Quadro 6 apresenta a escala de aptidão em técnicas e ferramentas para detecção de vulnerabilidades que foram avaliadas no questionário, sendo cada item categorizado na escala ordinal.

Quadro 6 – Escala de níveis de habilidades nas técnicas e ferramentas de segurança de software

Escala	Descrição dos níveis de habilidades
Baixo	Não possui experiência de trabalho nesta área.
Básico	Possui nível de experiência adquirida em cursos. Necessita de ajuda ao executar as tarefas.
Moderado	É capaz de completar com sucesso as tarefas nessa área. Pode solicitar o auxílio de um especialista, mas consegue executar as habilidades de forma independente.
Alto	É capaz de executar as tarefas associadas à área sem auxílio de terceiros. Pode ser reconhecido dentro da organização como um consultor imediato.
Especialista	É conhecido como um especialista. Possui capacidade de fornecer orientação, solucionar problemas e responder a perguntas relacionadas com esta área.

Fonte: adaptado de Oyetooyan, Cruzes e Gilje (2016)

6 - Descrição das características da implantação: foram definidas questões abertas sobre as características da implantação das técnicas e ferramentas, como: a motivação de sua escolha, as estratégias adotadas, os detalhes, as dificuldades e limitações encontradas e as lições aprendidas na implantação das técnicas e ferramentas.

No instrumento foram consideradas técnicas e ferramentas que não são exclusivamente de segurança, mas que podem ser adaptadas, como por exemplo: a revisão de código e as ferramentas de análise estática e dinâmica de código. A equipe pode utilizá-las para verificar a qualidade do código e a conformidade com padrão de boas práticas, detectando complexidades e mau cheiro de código (FOWLER; BECK, 1999).

Diante disso, as técnicas e ferramentas são escolhidas de forma voluntária pelas equipes ágeis, logo, os membros utilizam as que melhor se adequam às necessidades dos projetos e às suas atividades.

Mais detalhes sobre o instrumento desenvolvido poderão ser consultados no Apêndice A.

6.3.3 Coleta de dados

A seguinte estratégia foi utilizada para a coleta de dados:

1) Aplicação do instrumento desenvolvido na seção 5.3.2 por meio de aplicação de questionário e entrevistas com os gerentes das equipes de desenvolvimento de software e aplicação de questionário aos demais sujeitos da pesquisa, conforme apresentados na Seção 5.3.1.

2) A partir das respostas dos questionários e da entrevista, foi possível descrever e identificar as informações de diferentes níveis para as mesmas técnicas e ferramentas para detecção de vulnerabilidades em desenvolvimento ágil de software, levantando o estado da prática das técnicas e ferramentas de segurança, as motivações de implantação, as estratégias utilizadas, as dificuldades encontradas, as lições aprendidas e os benefícios obtidos com a implantação.

A partir da realização das entrevistas e da aplicação de questionários foi possível analisar o estado da prática dos membros e gerentes de equipes que utilizam métodos ágeis em relação à utilização de técnicas e ferramentas para detecção de vulnerabilidades.

6.3.4 Tratamento dos dados

Para o tratamento dos dados coletados nas questões abertas e entrevistas, foi utilizada a técnica de análise de conteúdo.

De acordo com a definição de Bardin (1991), análise de conteúdo é um conjunto de técnicas de análise das comunicações, que utiliza procedimentos sistemáticos e objetivos de descrição do conteúdo das mensagens. Assim, proporcionam o levantamento de indicadores, quantitativos ou não, permitindo a realização de inferência de conhecimento.

A análise de conteúdo é composta por três etapas: 1. Pré-análise; 2. Exploração do material; 3. Tratamento dos resultados, a inferência e a interpretação. A primeira etapa consiste na definição dos procedimentos a serem seguidos, bem como a escolha dos documentos que serão submetidos à análise. A segunda etapa consiste na organização

e implementação das regras definidas na etapa anterior. A última etapa consiste no tratamento dos resultados e aplicações de técnicas estatísticas (BARDIN, 1991).

O procedimento padrão para a análise de conteúdo consiste na definição de categorias, ou seja, agrupamento de elementos com características comuns a fim de potencializar a análise e comparações entre grupos.

As categorias devem ser: 1. Exaustiva, ou seja, permite a inclusão de praticamente todos os elementos, entretanto, isso às vezes não é possível; 2. Mutuamente exclusivas, ou seja, cada elemento só pode pertencer a uma única categoria; 3. Objetivas, ou seja, definida de maneira precisa evitando possíveis dúvidas na distribuição dos elementos; 4. Pertinentes, ou seja, adequadas ao objetivo da pesquisa (VERGARA, 2005).

Três abordagens podem ser selecionadas para a definição das categorias do procedimento padrão da análise de conteúdo: aberta, fechada e mista.

Na primeira, as categorias são definidas durante o andamento da pesquisa, permitindo a alteração do conteúdo até o momento de obtenção do conjunto final. Na segunda, as categorias são definidas *a priori*, ou seja, no início da pesquisa, juntamente com a revisão da literatura. Porém nesta grade alguns elementos podem ficar fora da categorização, visto que esta grade é rígida e não possui a flexibilidade da primeira.

A última abordagem incorpora a mistura das duas anteriores, ou seja, as categorias são definidas *a priori*, entretanto, elas são mutáveis, podendo ser alteradas no decorrer da coleta de dados. (VERGARA, 2005).

As etapas definidas por Vergara (2005) para coleta, análise, tratamento dos dados e as técnicas de análise de conteúdo que foram utilizadas nesta pesquisa são as seguintes:

Coleta dos dados: os dados foram coletados por meio de entrevistas semi-estruturadas e questionários, utilizando o instrumento desenvolvido.

Gravação de entrevistas: todas as entrevistas realizadas foram gravadas a partir da permissão do entrevistado por meio da ferramenta InqScribe³.

Transcrição das entrevistas gravadas: após a gravação das entrevistas, o conteúdo gerado foi transcrito no formato de um discurso. A partir dos textos transcritos, técnicas de análise de conteúdo foram aplicadas.

Grade para análise: a abordagem definida para análise foi a aberta, ou seja, as categorias de análise foram definidas durante o andamento da pesquisa.

³ <https://www.inqscribe.com/>

Categoria de análise: a partir da definição de utilização da abordagem aberta para a análise de conteúdo, foram definidas diferentes categorias com base nas respostas obtidas das entrevistas e das questões abertas, verificando esporadicamente sua conformidade com os objetivos da pesquisa.

Unidades de análise: as unidades de análise definidas foram frases e parágrafos, ou seja, foram analisados frases e parágrafos de cada entrevistado respondente e classificados de acordo com as categorias definidas.

Análise de conteúdo: após as definições das etapas anteriores, foi conduzida a análise de conteúdo apoiada em procedimentos interpretativos e com base nas categorias e unidades de análise definidas, bem como a situação problema e os objetivos desta pesquisa.

Já para a análise dos dados referentes as perguntas fechadas, foi aplicada a estatística descritiva.

6.4 Considerações finais

Este capítulo tratou dos procedimentos metodológicos utilizados nesta dissertação como: a relação entre objetivos, métodos e resultados; as informações sobre o *survey*; os sujeitos da pesquisa; o instrumento; a coleta dos dados; e o tratamento dos dados obtidos. O próximo capítulo apresenta a análise dos resultados.

7 Análise dos resultados

Este capítulo apresenta os resultados da aplicação do instrumento descrito no capítulo anterior. Ele consiste em questionários e entrevistas relacionadas ao uso de técnicas e ferramentas para detecção de vulnerabilidades em desenvolvimento ágil de software. Para cada um dos participantes deste *survey* foram analisados os seguintes aspectos: (1) características dos membros e gerentes de equipes de desenvolvimento ágil de software; (2) características dos projetos em que atuam; (3) técnicas e ferramentas utilizadas por membros e gerência nos projetos; (4) grau de implantação e ocorrência de benefícios na visão da gerência; (5) aptidão, interesse e necessidade de treinamento dos membros e gerência nas técnicas e ferramentas; e (6) características da implantação.

7.1 Resultados obtidos com o survey

A seguir, serão apresentados os resultados obtidos com o *survey*, de acordo com os objetivos gerais e específicos definidos na Seção 1.3.

7.1.1 Características dos membros das equipes de desenvolvimento ágil de software

Na Tabela 1, são apresentadas as características dos membros da equipe de desenvolvimento ágil de software, como função do membro na equipe e anos de atuação em determinada função. Um mesmo membro da equipe pode desempenhar mais de uma função, logo, o cálculo da porcentagem é baseado no total de respostas obtidas, 144, sendo que responderam ao questionário 110 membros.

Tabela 1 – Função dos respondentes e anos de atuação

		Quantidade %
Função na equipe	Desenvolvedor	39,58
	Testador	14,58
	Analista de requisitos	10,42
	Arquiteto de software	9,72
	<i>Agile coach</i>	8,33
	Especialista em segurança	7,64
	<i>Scrum master</i>	4,17
	Outros	5,56
	Total	100%
Anos na atual função	5 anos a 10 anos.	40,00
	2 anos a 5 anos.	38,18
	0 a 1 ano.	14,55
	Acima de 10 anos.	7,27
	Total	100%

Fonte: Lígia Santos, 2017

Nota-se que a função de desenvolvedor é a que possui maior percentual de grau de atuação, 39,58%. Em seguida, as funções de testador, analista de requisitos e arquiteto de software possuem, respectivamente, 14,58%, 10,42% e 9,72%. Já as funções de *agile coach*, especialista em segurança e *scrum master* possuem, respectivamente, 8,33%, 7,64% e 4,17%. Dessa forma, é possível observar que as principais funções abordadas pela pesquisa são as que possuem maior percentual e as demais funções são as que possuem menor percentual. Isso pode ser explicado pelo fato de a maioria das equipes participantes terem seus membros atuando em funções tradicionais de desenvolvimento de software, mesmo adotando métodos ágeis.

A função de *agile coach* vem sendo incorporada pelas equipes pelo fato de ter um membro designado como incentivador, facilitador e orientador que auxilia na difusão das práticas ágeis e solução de possíveis problemas durante o projeto. É uma função específica de métodos ágeis, assim como *scrum master* que incentiva a realização dos eventos Scrum, intermedia a comunicação entre equipe e cliente e atendimento ao escopo e custo do projeto.

Um especialista em segurança é uma função pouco incorporada nas equipes. Isso pode ser explicado pela falta de incentivo da organização, pelas dificuldades em obter treinamento ou pela dificuldade em encontrar e incorporar especialistas em segurança em equipes ágeis.

Ainda na Tabela 1 é possível notar que 40% dos membros das equipes possuem de 5 a 10 anos de atuação em desenvolvimento de software. Em seguida, membros com 2 a 5 anos e 0 a 1 ano de experiência possuem, respectivamente, 38,18% e 14,55%. Já membros com mais de 10 anos de experiência obtiveram 7,27%.

Outras funções somam 5,56% das respostas obtidas. Elas são funções relacionadas à gestão e liderança, como gestor e coordenador de TI ou relacionadas à qualidade de software e ciência dos dados como, respectivamente, analistas de qualidade e cientista de dados.

7.1.2 Características dos projetos

Na Tabela 2, pode-se observar os métodos ágeis utilizados pelas equipes, tecnologias utilizadas, metodologias de implantação das técnicas e ferramentas, tamanho da equipe que atuou na implantação e responsáveis por segurança durante o desenvolvimento de software. Assim, uma mesma equipe pode desenvolver software utilizando mais de um método ágil, atuar com diversas tecnologias e adotar mais de uma metodologia de implantação de segurança. Logo, o cálculo da porcentagem é baseado no total de respostas obtidas para cada uma das questões.

Tabela 2 – Métodos ágeis utilizados, tecnologias adotadas, metodologias, equipe e responsável pela implantação de segurança

		Quantidade %
Métodos ágeis utilizados	Scrum	41,22
	Kanban	26,94
	Extreme Programming (XP)	11,43
	Lean Software Development	7,76
	Feature Driven Development (FDD)	4,08
	Agile Unifield Process (AUP)	2,45
	Crystal Methods	2,04
	Dynamics Systems Development Method (DSDM)	1,22
	Outros	2,86
	Total	100%
Tecnologia utilizada	Web	47,89
	Mobile	21,60
	E-commerce	15,02
	Rede	9,86
	Desktop	4,23
	Outras	1,41
	Total	100%
Metodologia de implantação	Nenhuma	33,87
	Metodologia própria	30,65
	ITIL	16,94
	CMMI	8,87
	MPS.br	5,65
	Não sei	1,61
	Outros	2,42
	Total	100%
Tamanho da equipe	De 0 a 5 pessoas.	52,73
	De 6 a 10 pessoas.	23,64
	Acima de 10 pessoas.	10,91
	Em branco.	12,73
	Total	100%
Responsável por segurança	Desenvolvedor	23,08
	Sem atribuição formal para essa responsabilidade	23,08
	Especialista em segurança	17,95
	Arquiteto de software	16,67
	Analista de requisitos	9,62
	Testador	9,62
	Total	100%

Fonte: Lígia Santos, 2017

A Tabela 2 mostra os métodos ágeis utilizados pelos membros de equipes. Uma mesma equipe pode desenvolver software com mais de um método ágil, por isso, foram obtidas 245 respostas, sendo 110 o total de respondentes. O cálculo da porcentagem é baseado no total de respostas. Dessa forma, é possível observar que o *framework* “Scrum” (SCHWABER, 2013), obtendo 41,22% de uso, é o mais popular. Em seguida, “Kanban” (SKARIN, 2015), “Extreme Programming (XP)” (BECK, 2000) e “Lean” (POPPENDIECK; POPPENDIECK, 2003) possuem, respectivamente, 26,94%, 11,43% e 7,76%. A grande popularidade de “Scrum” é porque ele pode ser usado conjuntamente com outros métodos ágeis, utilizando apenas os eventos que melhor se adequam aos projetos e contexto da organização, ou ainda os mais consolidados. Os métodos ágeis menos utilizados, como: “Feature Driven Development (FDD)” (PALMER; FELSING, 2002), “Agile Unified Process (AUP)” (AMBLER, 2006), “Crystal Methods” (COCKBURN, 2004) e “Dynamic Systems Development Method

(DSDM)” (DAVIES, 2004) possuem, respectivamente, 4,08%, 2,45%, 2,04% e 1,22%. Outros métodos ágeis utilizados somam 2,86% das respostas obtidas.

Os métodos ágeis podem ser combinados com ferramentas para facilitar a sua adoção ou ainda a utilização de diversas práticas ágeis isoladas. O motivo de se utilizar métodos ágeis e ferramentas pode ser explicado pela possibilidade de contribuição na realização das práticas ágeis, facilidade em acompanhar seus eventos e visualizar melhor os artefatos produzidos nas iterações e comparar resultados. Já o motivo de se utilizar práticas ágeis independente de seguir um método ágil específico pode ser explicado pela possibilidade de se utilizar as práticas ágeis mais consolidadas, as mais aceitas pelos membros das equipes e as que se adequam melhor ao contexto de determinado projeto. A motivação para a utilização de práticas ágeis isoladas também pode estar relacionada a dificuldades ou limitações dos membros das equipes em seguir todos eventos ou falta de aderência das práticas de um único método ágil nos projetos em que atuam.

Em relação às tecnologias adotadas pelos membros de equipes, a maioria atua em projetos de tecnologia *web* e *mobile* obtendo 47,89% e 21,60%, respectivamente. Eles também atuam em projetos de *e-commerce*, redes e *desktop* obtendo, respectivamente, 15,02%, 9,86% e 4,23% das respostas. Cada tecnologia possui sua importância, porém a alta porcentagem na atuação em projetos *web* e *mobile* pode ser explicada pelo fato de que a maioria das organizações investe em aplicativos *mobile* e *web*, por serem tecnologias atuais e que são utilizadas conjuntamente. Já a tecnologia de *e-commerce* é importante em contextos em que o foco é realizar transações financeiras por meio de plataformas ou dispositivos eletrônicos.

Outras tecnologias somam 1,41% de utilização pelos respondentes. Elas são tecnologias *mainframe*, computação em nuvem e análise de dados.

No que concerne as metodologias utilizadas para implantação das técnicas e ferramentas para detecção de vulnerabilidades, 33,87% dos membros que participaram desta pesquisa não utilizaram metodologias. 30,65% deles utilizam metodologias próprias para realizar a implantação. Elas podem ser conjuntos de boas práticas de implantação existentes ou práticas isoladas que a equipe julga ser útil para o momento ou a mais adequada ao contexto do projeto ou da organização. O *framework* ITIL obteve 16,94% das respostas. O ITIL (*Information Technology Infrastructure Library*) contribui para a implantação e gerenciamento de serviços de Tecnologia da Informação (FREITAS, 2013). O CMMI (*Capability Maturity Model Integration*) é um modelo composto por práticas para ob-

tenção e avaliação de maturidade em processos de software (SEI, 2017). Ele pode ser utilizado conjuntamente com atividades de segurança, orientando a equipe durante a sua implantação.

Assim, 8,87% dos membros de equipes utilizam o CMMI como metodologia para auxílio à implantação das técnicas e ferramentas. O MPS.br (Melhoria do Processo de Software Brasileiro) é um modelo de melhoria e qualidade nos processos, o qual pode incorporar segurança no processo de desenvolvimento de software (SOFTEX, 2003). Dessa forma, foi a metodologia utilizada para implantação das técnicas e ferramentas para 5,65% dos respondentes.

Já 1,61% dos respondentes não sabem quais foram as metodologias utilizadas para auxiliar a implantação e 2,42% utilizam outras metodologias.

7.2 *Técnicas e ferramentas para detecção de vulnerabilidades*

A seguir, serão apresentadas as técnicas e ferramentas utilizadas em cada fase do ciclo de vida ágil pelos membros de equipes, bem como o seu grau de implantação e benefícios obtidos.

7.2.1 Identificação e descrição das técnicas e ferramentas adotadas pelos membros de equipes ágeis

A Tabela 3 apresenta a relação das 18 técnicas e ferramentas para detecção de vulnerabilidades listadas de acordo com a revisão bibliográfica realizada. Elas foram classificadas nas seguintes fases: análise de requisitos e modelagem, *design*, codificação, testes e produção. A partir dessa classificação, foi possível analisar a porcentagem do grau de uso atual de cada uma das técnicas e ferramentas, bem como sua utilização por fase do ciclo de vida de desenvolvimento de software. O cálculo está baseado nas 110 respostas obtidas.

Tabela 3 – Uso atual de técnicas e ferramentas para detecção de vulnerabilidades nos projetos

Fase	Técnicas e ferramentas	Sim %	Não %	Sim* %	Não* %
AR e mod.	Especialista em segurança	19,09	66,36	36,36	49,09
	Requisitos de segurança	21,82	63,64		
	<i>Abuse/Misuse Case/Stories</i>	17,27	68,18		
	Modelagem de ameaças	13,64	71,82		
	Modelagem de riscos	16,36	69,09		
	Ferramenta de modelagem de riscos e ameaças	13,64	71,82		
Design	<i>Design</i> de segurança	21,82	63,64	40,00	45,45
	Contramedidas de segurança	20,00	65,45		
	Avaliação de vulnerabilidades	32,73	52,73		
Codificação	Codificação segura	44,55	40,91	56,36	29,09
	Ferramenta de análise estática de código	34,55	50,91		
	Revisão de segurança de código	30,91	54,55		
	Ferramenta de revisão de código	41,82	43,64		
Testes	Ferramenta de análise dinâmica de código	31,82	53,64	41,82	43,64
	Teste de penetração	11,82	73,64		
	<i>Fuzz testing</i>	18,18	67,27		
	Teste baseado em riscos	17,27	68,18		
Prod.	Gestão de respostas a incidentes	21,82	63,64	21,82	63,64

AR e mod = Análise de requisitos e modelagem — Prod. = Produção — * = Média do uso por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

A fase de codificação possui maior grau de utilização de técnicas e ferramentas se comparadas às demais, alcançando 56,36% de utilização. Esse alto grau pode ser explicado pelo fato de a fase de codificação possuir técnicas e ferramentas mais conhecidas e acessíveis, com materiais e treinamentos disponíveis. Dentre as técnicas e ferramentas que compõem essa fase, a codificação segura e a ferramenta de revisão de código são as mais utilizadas com, respectivamente, 44,55% e 41,82%. As menos utilizadas são as técnicas de revisão de segurança de código e ferramenta de análise estática de código com, respectivamente, 30,91% e 34,55%. O percentual de grau de uso das técnicas e ferramentas variam por conta dos diversos fatores como a necessidade, o contexto dos projetos, o custo, o prazo, a capacitação dos membros de equipes e incentivo da organização em implantar segurança em seus projetos.

A fase de testes é a segunda que possui técnicas e ferramentas mais utilizadas, alcançando 41,82% de utilização. Mesmo possuindo técnicas e ferramentas conhecidas, como ferramenta de análise dinâmica de código, teste de penetração, *fuzz testing* e teste

baseado em riscos, a porcentagem é baixa se comparada à fase de codificação. Esse dado pode ser explicado pelo fato de algumas técnicas e ferramentas, como o teste baseado em riscos, *fuzz testing* e o teste de penetração, não serem muito utilizados se comparados a outras técnicas e ferramentas de testes ou ainda pela possibilidade de terem um custo mais elevado, serem mais complexas e por dificuldades na aquisição de manuais ou treinamentos. A ferramenta mais utilizada nessa fase é a análise dinâmica de código com 31,82%.

O grau de uso na fase de *design* encontra-se em 40%. Dentre as técnicas e ferramentas que compõem essa fase, a técnica de avaliação de vulnerabilidades e *design* de segurança são as que possuem maiores percentuais com, respectivamente, 32,73% e 21,82%. Contramedidas de segurança é a técnica menos utilizada, possuindo 6,45%.

Na fase de análise de requisitos e modelagem a utilização atual chega a 36,36%. Esse percentual de grau de uso deve-se à falta de adoção de técnicas e ferramentas pela equipe, por não ter conhecimento de sua existência, por negligenciar segurança durante as fases iniciais dos projetos ou ainda pela pressão na entrega do produto. A técnica de requisitos de segurança e especialistas em segurança são as que possuem maiores percentuais com, respectivamente, 21,82 e 19,09%.

Finalmente, a fase de produção é a que possui menor percentual de grau de utilização com 21,81%. As respostas em branco somam 14,55% das respostas obtidas.

A partir desses dados foi possível concluir que as técnicas e ferramentas para detecção de vulnerabilidades são pouco utilizadas nos projetos visto que, dentre as porcentagens de uso obtidas, nenhuma obteve percentual igual ou acima de 50%. As técnicas e ferramentas que mais se aproximaram foi a codificação segura, ferramenta de revisão de código, ferramenta de análise estática de código, avaliação de vulnerabilidades e ferramenta de análise dinâmica de código com, respectivamente, 44,55%, 41,82%, 34,55%, 32,73% e 31,82%.

7.2.2 Grau de implantação das técnicas e ferramentas e benefícios associados

A Tabela 4 apresenta a relação das 18 técnicas e ferramentas para detecção de vulnerabilidades. Para cada uma delas, foram definidas quatro questões relacionadas a ocorrência da sua implantação e quatro questões relacionadas a ocorrência de benefícios. Os gerentes apontaram a ocorrência ou não de implantação de determinada técnica e

ferramenta, bem como a ocorrência ou não de benefícios para essas. O cálculo está baseado nas 15 respostas obtidas.

Tabela 4 – Grau de implantação e ocorrência de benefícios na implantação das técnicas e ferramentas para detecção de vulnerabilidades

Fases	A.R e mod.		<i>Design</i>		Codificação		Testes		Produção	
Técnicas e ferramentas	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %
Especialista em segurança	71,67	68,33								
Requisitos de segurança	36,67	41,67								
<i>Abuse/Misuse Case/Stories</i>	15,00	48,33								
Modelagem de ameaças	38,33	50,00								
Modelagem de riscos	40,00	53,33								
Ferramenta de modelagem de riscos e ameaças	28,33	33,33								
<i>Design</i> de segurança			26,67	30,00						
Contramedidas de segurança			38,33	36,67						
Avaliação de vulnerabilidades			43,33	38,33						
Codificação segura					55,00	58,33				
Ferramenta de análise estática de código					45,00	46,67				
Revisão de segurança de código					38,33	36,67				
Ferramenta de revisão de código					40,00	45,00				
Ferramenta de análise dinâmica de código							23,33	36,67		
Teste de penetração							13,33	13,33		
<i>Fuzz testing</i>							13,33	13,33		
Teste baseado em riscos							35,00	35,00		
Gestão de respostas a incidentes									43,33	43,33

A.R e mod. = Análise de requisitos e modelagem — Impl. = Implantação — Ben. = Benefícios

Fonte: Lígia Santos, 2017

Dentre as técnicas e ferramentas listadas é possível observar que muitas delas possuem baixo grau de implantação, porém, seus benefícios são mais altos. Isso pode ser explicado pelo contexto e necessidade do projeto. Um projeto não necessita de inúmeras técnicas e ferramentas para detecção de vulnerabilidades implantadas. Implantar técnicas e ferramentas desnecessárias contribui para a não ocorrência de benefícios visto que poderão surgir problemas de onerosidade durante o desenvolvimento, já que os gerentes deverão propor e organizar treinamentos e incentivar o seu uso, elaborar e divulgar a documentação das técnicas e ferramentas e solucionar possíveis problemas com custo e prazo do projeto.

Dessa forma, é possível analisar que na fase de análise de requisitos e modelagem, algumas técnicas e ferramentas obtiveram baixo grau de implantação e alto grau de benefícios. Essas ocorrências podem ser observadas, respectivamente, nas técnicas de requisitos de segurança, que obteve grau de implantação de 36,67% e grau de benefícios 41,67%; *abuse/misuse cases/stories* com grau de implantação de 15% e grau de benefícios 48,33%; modelagem de ameaças com grau de implantação de 38,33% e grau de benefícios em 50%; modelagem de riscos com 40% de grau de implantação e 53,33% de grau de implantação e ferramentas de modelagem de riscos e ameaças com 28,33% de grau de implantação e 33,33% de grau de benefícios.

Na fase de *design*, a técnica de *design* seguro obteve baixo grau de implantação e alto grau de benefícios, pontuando 26,67% de grau de implantação e 30% de grau de benefício. Dentre as técnicas e ferramentas que compõem a fase de codificação, a técnica de codificação segura obteve 55% de grau de implantação e 58,33% de grau de benefícios; ferramenta de análise estática de código obteve 45% de grau de implantação e 46,67% de grau de benefícios e finalmente a ferramenta de revisão de código, obteve 40% de grau de implantação e 45% de grau de benefícios.

Para técnicas e ferramentas que compõem a fase de testes, a técnica de ferramenta de revisão de código obteve 23,33% de grau de implantação e 36,67% de grau de benefícios. O restante das técnicas obtiveram o mesmo grau de implantação e benefícios. Já na fase de produção, o grau de implantação e benefício foi o mesmo para a gestão de respostas a incidentes.

Estes dados podem ser explicados pelo fato de os membros de equipes e seus gerentes implantarem apenas algumas das questões. Essa ocorrência ou não de implantação é influenciada por fatores como a exigência do cliente, recomendação de algum membro, baixo custo, facilidade de entendimento e implantação, entre outros. Essa ocorrência

ou não de implantação está associada a um grau de benefício. Sendo assim, o grau de implantação e ocorrência de benefícios é relativo, pois como já foi dito, a equipe pode implantar nenhuma ou até quatro questões de implantação e obter algum grau de benefício.

Entretanto, nem sempre ter alto grau de implantação nas questões de técnicas e ferramentas para detecção de vulnerabilidades nos projetos proporcionará alto grau de benefícios. Exemplos dessa ocorrência são os especialistas em segurança que obtiveram 71,67% de grau de implantação e 68,33% de grau de benefícios; contramedidas de segurança com 38,33% de grau de implantação e 36,67% de grau de benefícios; avaliação de vulnerabilidades com 43,33% de grau de implantação e 38,33% de grau de benefícios e revisão de segurança de código com 38,33% de grau de implantação e 36,67% de grau de benefícios.

Isso pode ser explicado pelo fato de algumas técnicas e ferramentas implantadas serem ineficazes ao problema proposto, pela possibilidade de não estarem completamente implantadas, pelo excesso de técnicas e ferramentas ou sua má utilização. Sendo assim, o baixo grau de benefícios pode ser causado por esses aspectos.

A Tabela 5 apresenta a relação das 18 técnicas e ferramentas de detecção de vulnerabilidades, agrupadas em cinco fases, conforme o ciclo de vida de desenvolvimento de software. As fases são: análise de requisitos e modelagem, *design*, codificação, testes e produção. Para cada uma das técnicas e ferramentas agrupadas foram definidas quatro questões relacionadas a sua implantação e quatro questões relacionadas aos seus benefícios. As possíveis pontuações dessas questões são: 0 (não implantado), 25% (uma questão implantada), 50% (duas questões implantadas), 75% (três questões implantadas) e 100% (todas questões implantadas). Em seguida, foi calculada a média da ocorrência de implantação das questões e benefícios obtidos, considerando cada uma das fases.

Assim, os gerentes apontaram a ocorrência da implantação de questões e benefícios, relacionados à intenção da implantação total de determinada técnica e ferramenta. O cálculo do grau de implantação e benefícios está baseado nas 15 respostas obtidas por meio de um questionário para gerentes.

Tabela 5 – Grau de implantação e ocorrência de benefícios na implantação das técnicas e ferramentas para detecção de vulnerabilidades por fase

Fases	A.R e mod.		<i>Design</i>		Codificação		Testes		Produção	
Técnicas e ferramentas	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %	Impl. %	Ben. %
Especialista em segurança	38,33	49,17								
Requisitos de segurança										
<i>Abuse/Misuse Case/Stories</i>										
Modelagem de ameaças										
Modelagem de riscos										
Ferramenta de modelagem de riscos e ameaças			36,11	35,00						
<i>Design</i> de segurança										
Contra medidas de segurança										
Avaliação de vulnerabilidades					44,58	46,67				
Codificação segura										
Ferramenta de análise estática de código										
Revisão de segurança de código										
Ferramenta de revisão de código							21,25	24,58		
Ferramenta de análise dinâmica de código										
Teste de penetração										
<i>Fuzz testing</i>										
Teste baseado em riscos									43,33	43,33
Gestão de respostas a incidentes										

A.R e mod. = Análise de requisitos e modelagem — Impl. = Implantação — Ben. = Benefícios
 Fonte: Lígia Santos, 2017

Logo, a gerência deverá avaliar posteriormente se as técnicas e ferramentas são as melhores para o projeto, se suas questões de implantação foram implantadas e quais são os seus benefícios associados. Manter técnicas ou ferramentas desnecessárias no projeto podem causar prejuízos, tanto em custo, quanto em desempenho.

7.3 *Aptidão, interesse e necessidade de treinamento*

Neste capítulo, serão tratadas as aptidões, interesses e necessidades de treinamento em técnicas e ferramentas para detecção de vulnerabilidades.

7.3.1 Aptidão dos membros de equipes ágeis em técnicas e ferramentas

A seguir, serão discutidos os extremos da escala utilizada especialista e baixa aptidão nas técnicas e ferramentas.

A Tabela 6 apresenta as aptidões dos 110 membros de equipes que participaram do *survey*. Em relação ao grau de aptidão especialista, na fase de produção, a gestão de respostas a incidentes possui maior porcentagem com 4,55%. Essa porcentagem está ligada ao fato de a gestão de respostas a incidentes não ser algo complexo se comparadas as outras técnicas e ferramentas mais refinadas. Tal técnica não exige inúmeros recursos e nem é necessário uma equipe grande. Para implantar uma gestão de respostas a incidentes eficiente é necessário inseri-la na rotina da organização, identificando e mapeando possíveis incidentes, definindo planos de contingência e os monitorando. Assim, a equipe deve ser formada por um grupo de pessoas que saibam gerenciar incidentes de segurança, mantendo um plano de contingência ativo, eficiente e eficaz.

As fases de *design* e codificação possuem, respectivamente, 4,24 % e 4,09% das respostas obtidas. Essas porcentagens obtidas nas fases de *design* e codificação possuem grau alto se comparadas às outras fases pelo fato de existirem diversas técnicas e ferramentas para detecção de vulnerabilidades nas fases de *design* e codificação. Na fase de *design* existem técnicas de *design* de segurança, na qual são avaliadas as possíveis vulnerabilidades, modeladas árvores de riscos e ameaças, definidas arquiteturas de segurança e realizadas contramedidas de segurança para os riscos e ameaças modelados. A técnica de avaliação de vulnerabilidades é a que possui maior grau de aptidão com 5,45% e as técnicas de

design de segurança e contramedidas de segurança possuem 3,64%. Para realizar um *design* seguro, não é necessário existirem especialistas em segurança, apenas interesse do cliente e iniciativa da equipe em modelar vulnerabilidades e evitar incidentes de segurança. Já na fase de codificação, dentre as técnicas e ferramentas que compõem tal fase, ferramenta de revisão de código, análise estática de código e codificação segura são as que possuem maior percentual no grau de aptidão especialista, com 7,27%, 4,55% e 2,73%, respectivamente.

A fase de testes possui 3,41% no grau de aptidão especialista. Ela é composta por técnicas e ferramentas como: análise dinâmica de código, teste de penetração, *fuzz testing* e teste baseado em riscos. Dentre elas, o que possui maior percentual de especialistas é o teste de penetração com 5,45% das respostas e as demais possuem 1,82%.

Finalmente, a fase de análise de requisitos e modelagem possui 3,33% das respostas obtidas para o grau especialista. Dentre as técnicas que compõem tal fase, especialista em segurança, modelagem de ameaças e riscos são as que possuem maior percentual, com 5,45%, 4,55% e 3,64%, respectivamente. Especialista em segurança é alguém capaz de definir políticas de segurança, gerenciar as atividades e incentivar seu uso dentro do projeto. Na fase de análise de requisitos, ele é responsável por incentivar a equipe a definir estratégias de segurança a serem utilizadas durante todo o projeto. A modelagem de riscos e ameaças contribuem para a identificação de possíveis ameaças nos componentes do software e a sua probabilidade de ocorrência. Tais técnicas e ferramentas devem ser melhoradas a cada iteração para adquirirem mais detalhes sobre possíveis riscos e ameaças, além de servirem de base para a fase de codificação.

Os graus de aptidão baixos e básicos estão, respectivamente, nas fases de testes e *design*, com 35% e 27,27%. Mesmo a fase de testes sendo composta por técnicas e ferramentas conhecidas, ela possui um percentual alto de membros que possuem baixa aptidão. Isso pode ser explicado pelo fato de as equipes possivelmente terem implantado técnicas e ferramentas para detecção de vulnerabilidades em outras fases do ciclo de vida, por exemplo, durante a fase de codificação ou *design*, por não utilizarem as técnicas e ferramentas listadas, por não terem conhecimento o suficiente para realizarem testes com tais técnicas e ferramentas, ou ainda por falta de interesse ou necessidade. *Fuzz testing* é a técnica que possui maior percentual de baixa aptidão com 45,45%. Já para a fase de *design*, a técnica de contramedidas de segurança possui o maior percentual de grau básico de aptidão com 28,18%. Em todas as fases, as respostas em branco somam 14,55%.

Tabela 6 – Aptidão dos membros nas técnicas e ferramentas

Fases	Técnicas e ferramentas	Bx. %	Bas. %	Mod. %	A %	Esp. %	Bx.* %	Bas.* %	Mod.* %	A* %	Esp.* %
AR e mod.	Especialista em segurança	32,73	28,18	13,64	5,45	5,45	33,33	26,97	15,15	6,67	3,33
	Requisitos de segurança	21,82	33,64	16,36	10,91	2,73					
	<i>Abuse/Misuse Case/Stories</i>	37,27	25,45	14,55	5,45	2,73					
	Modelagem de ameaças	39,09	25,45	14,55	1,82	4,55					
	Modelagem de riscos	28,18	23,64	20,91	9,09	3,64					
	Ferramenta de modelagem de riscos e ameaças	40,91	25,45	10,91	7,27	0,91					
Design	<i>Design</i> de segurança	36,36	27,27	10,91	7,27	3,64	32,42	27,27	11,52	10,00	4,24
	Contrameditadas de segurança	34,55	28,18	12,73	6,36	3,64					
	Avaliação de vulnerabilidades	26,36	26,36	10,91	16,36	5,45					
Codificação	Codificação segura	20,91	22,73	24,55	14,55	2,73	22,50	24,55	23,64	10,68	4,09
	Ferramenta de análise estática de código	17,27	26,36	28,18	9,09	4,55					
	Revisão de segurança de código	33,64	25,45	17,27	7,27	1,82					
	Ferramenta de revisão de código	18,18	23,64	24,55	11,82	7,27					
Testes	Ferramenta de análise dinâmica de código	20,91	27,27	23,64	9,09	4,55	35,00	25,00	14,55	7,50	3,41
	Teste de penetração	36,36	24,55	10,00	9,09	5,45					
	<i>Fuzz testing</i>	45,45	25,45	8,18	4,55	1,82					
	Teste baseado em riscos	37,27	22,73	16,36	7,27	1,82					
Prod.	Gestão de respostas a incidentes	22,73	21,82	25,45	10,91	4,55	22,73	21,82	25,45	10,91	4,55

A.R = Análise de Requisitos — Bx. = Baixa — Bas. = Básica — Mod. = Moderada — A = Alta — Esp. = Especialista — * = Média da aptidão por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

7.3.2 Interesse dos membros de equipes ágeis nas técnicas e ferramentas

A Tabela 7 apresenta a relação de técnicas e ferramentas nas quais os membros de equipes possuem interesses. A fase de codificação é a que possui maior percentual de grau muito interessado com 17,27. Dentre as técnicas e ferramentas que compõem essa fase, as técnicas de codificação segura e revisão de segurança de código possuem os maiores percentuais 23,64% e 16,36%, respectivamente.

A fase de testes possui percentual de 15%. As técnicas e ferramentas de *fuzz testing* e análise dinâmica de código possuem os maiores percentuais de grau muito interessado com, respectivamente, 17,27% e 16,36%. As fases de *design* e produção possuem, respectivamente, 13,94% e 11,82%. Na fase de *design*, a avaliação de vulnerabilidades e o *design* de segurança são os que obtiveram maior percentual com, respectivamente, 18,18% e 13,64%. Já a fase de produção, a técnica de teste baseado em riscos obteve 11,82%.

Finalmente, a fase de análise de requisitos e modelagem obteve 8,64%. Dentre as técnicas e ferramentas dessa fase, as que obtiveram maior percentual, foram especialista em segurança e técnica de requisitos de segurança com 11,82%. Ainda 10% dos respondentes tem muito interesse em utilizar ferramentas de modelagem de riscos e ameaças.

Em relação ao grau não interessado, as fases de análise de requisitos e testes obtiveram os maiores percentuais com, respectivamente, 15,30% e 15,23%. Na primeira, as técnicas de *abuse/misuse cases/stories* e modelagem de ameaças obtiveram, respectivamente, 20% e 17,27%. Na segunda, as técnicas de teste de penetração e *fuzz testing* obtiveram os maiores graus de desinteresse. Esses valores podem ser explicados pelo fato de não haver necessidade em utilizar essas técnicas e ferramentas nos projetos em que os membros de equipes atuam, já que o percentual de utilização dessas técnicas e ferramentas são baixos, se comparadas às outras e seus graus de aptidão nessas mesmas técnicas e ferramentas são altos. Dessa forma, é possível inferenciar que a falta de uso e baixo grau de aptidão causa o desinteresse na técnica e ferramenta. Em branco somam 14,55% das respostas obtidas.

Tabela 7 – Interesse da equipe de desenvolvimento em técnicas e ferramentas para detecção de vulnerabilidades

Fases	Técnicas e ferramentas	NI %	LI %	Mod. %	I %	MI %	NI* %	LI* %	Mod.* %	I* %	MI* %
AR e mod.	Especialista em segurança	15,45	20,91	16,36	20,91	11,82	15,30	19,85	17,88	23,79	8,64
	Requisitos de segurança	11,82	19,09	15,45	27,27	11,82					
	<i>Abuse/Misuse Case/Stories</i>	20,00	20,91	20,91	18,18	5,45					
	Modelagem de ameaças	17,27	19,09	18,18	25,45	5,45					
	Modelagem de riscos	13,64	20,00	16,36	28,18	7,27					
	Ferramenta de modelagem de riscos e ameaças	13,64	19,09	20,00	22,73	10,00					
Design	<i>Design</i> de segurança	10,91	18,18	19,09	23,64	13,64	12,42	17,27	15,15	26,67	13,94
	Contra medidas de segurança	13,64	24,55	9,09	28,18	10,00					
	Avaliação de vulnerabilidades	12,73	9,09	17,27	28,18	18,18					
Codificação	Codificação segura	10,91	6,36	17,27	27,27	23,64	10,68	10,45	19,77	27,27	17,27
	Ferramenta de análise estática de código	12,73	16,36	19,09	22,73	14,55					
	Revisão de segurança de código	8,18	10,00	19,09	31,82	16,36					
	Ferramenta de revisão de código	10,91	9,09	23,64	27,27	14,55					
Testes	Ferramenta de análise dinâmica de código	12,73	7,27	22,73	26,36	16,36	15,23	15,91	17,27	22,05	15,00
	Teste de penetração	18,18	22,73	13,64	19,09	11,82					
	<i>Fuzz testing</i>	15,45	14,55	15,45	22,73	17,27					
	Teste baseado em riscos	14,55	19,09	17,27	20,00	14,55					
Prod.	Gestão de respostas a incidentes	13,64	16,36	16,36	27,27	11,82	13,64	16,36	16,36	27,27	11,82

A.R = Análise de Requisitos — NI = Não interessado — LI = Levemente interessado — Mod. = Moderadamente interessado — I = Interessado — MI. = Muito interessado — * = Média de interesse por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

A Tabela 8 agrupa as técnicas e ferramentas para detecção de vulnerabilidades, independente das fases do ciclo de vida de desenvolvimento de software. Esses conjuntos são compostos por técnicas e ferramentas das diferentes fases do ciclo de vida, o que pode contribuir para a ampliação do conhecimento na implantação de segurança de software durante todo o projeto.

O percentual de interesse em receber treinamento nesses conjuntos de técnicas e ferramentas é variado. Os conjuntos 2, 5 e 6 são os que possuem maior percentual de interesse em treinamento com, respectivamente, 58,18%, 55,45% e 55,45%. O conjunto 2 é composto pelas técnicas de modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes. O conjunto 5 é composto pela avaliação de vulnerabilidades, teste de penetração e *fuzz testing*. Já o conjunto 6 é composto pelas técnicas de teste baseado em riscos e revisão de segurança de código.

Essa variação no percentual é motivada pelo contexto do projeto em que os membros de equipes estão atuando, por interesse pessoal em adquirir conhecimento nas técnicas e ferramentas que compõem a fase, por exigência ou recomendação de segurança por parte da gerência e cliente ou ainda pela garantia da qualidade do software desenvolvido.

Tabela 8 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades

Interesse em receber treinamento	Sim %	Não %
1 - Especialista em segurança, requisitos de segurança e <i>abuse/misuse cases/stories</i> .	42,73	41,82
2 - Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	58,18	26,36
3 - <i>Design</i> de segurança, contramedidas de segurança e codificação segura.	50,00	34,55
4 - Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	51,82	32,73
5 - Avaliação de vulnerabilidades, teste de penetração e <i>Fuzz testing</i> .	55,45	29,09
6 - Teste baseado em riscos e revisão de segurança de código.	55,45	29,09

Fonte: Lígia Santos, 2017

7.3.3 Necessidade de treinamento dos membros de equipes ágeis nas técnicas e ferramentas

A Tabela 9 apresenta o grau de interesse dos membros de equipes em receber treinamento em um conjunto de técnicas e ferramentas para detecção de vulnerabilidades. Sendo assim, para o grau muito interessado, os conjuntos 6, 4 e 3 foram os que obtiveram maior percentual com, respectivamente, 27,27%, 25,45% e 23,64%. O conjunto 6 é composto pelas técnicas de teste baseado em riscos e revisão de segurança de código. O conjunto 4 é composto pelas técnicas e ferramentas de modelagem de ameaças e riscos, ferramentas de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código. Já o conjunto 3 é composto pelas técnicas de *design* seguro, contramedidas de segurança e codificação segura. Para o grau moderado, o conjunto 2, composto pelas técnicas de modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes, obteve 24,55%. O segundo maior percentual é do conjunto 5, composto pelas técnicas de avaliação de vulnerabilidades, teste de penetração e *fuzz testing* obteve 21,82%. Para os graus levemente interessado, os maiores percentuais foram dos conjuntos 2 com 10,91% e o conjunto 5 com 9,09%.

Para o grau não interessado, o conjunto 1 composto pelas técnicas e ferramentas de especialista em segurança, requisitos de segurança e *abuse/misuse cases/stories* obteve 26,36% e o conjunto 3 obteve 23,64%.

Tabela 9 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades

Grau de interesse em receber treinamento	NI. %	LI. %	Mod. %	MI. %	NS. %	Br. %
1 - Especialista em segurança, requisitos de segurança e <i>abuse/misuse cases/stories</i> .	26,36	7,27	15,45	20,00	0,00	30,91
2 - Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	19,09	10,91	24,55	20,00	0,91	24,55
3 - <i>Design</i> de segurança, contramedidas de segurança e codificação segura.	23,64	8,18	17,27	23,64	0,91	26,36
4 - Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	20,00	5,45	18,18	25,45	1,82	29,09
5 - Avaliação de vulnerabilidades, teste de penetração e <i>Fuzz testing</i> .	22,73	9,09	21,82	20,91	2,73	22,73
6 - Teste baseado em riscos e revisão de segurança de código.	21,82	5,45	17,27	27,27	4,55	23,64

NI. = Não interessado — LI. = Levemente interessado — Mod. = Moderadamente interessado — MI = Muito interessado — NS. = Não sei — Br. = Branco

Fonte: Lígia Santos, 2017

7.3.4 Características da implantação das técnicas e ferramentas

As características da implantação das técnicas e ferramentas para detecção de vulnerabilidades foram obtidas por meio de perguntas abertas para os 110 membros de equipes de desenvolvimento de software. As perguntas tratavam das motivações, estratégias, dificuldades, limitações e lições aprendidas durante esse processo. Todas as perguntas abertas tratadas nesta seção passaram pela técnica de análise de conteúdo conforme descrito na Seção 5. A Tabela 10 apresenta as motivações obtidas para a implantação das técnicas e ferramentas.

Tabela 10 – Motivação para a implantação de segurança no desenvolvimento ágil de software

Motivações para a implantação de segurança	Quantidade %
Mitigação de riscos e ameaças	22,73
Qualidade do software	17,27
Exigência de segurança de software pelo cliente	10,91
Melhoria na gestão da segurança nos projetos	1,82
Falta de segurança dos sistemas existentes	0,91
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	9,09
Não sei	5,45
Branco	31,82
Total	100%

Fonte: Lígia Santos, 2017

Ao serem questionados sobre as motivações da implantação de técnicas e ferramentas para detecção de vulnerabilidades no desenvolvimento ágil de software, 22,73% dos membros das equipes citaram a mitigação dos riscos e possíveis ameaças como maior motivação. Já para 17,27% dos membros das equipes, a qualidade do software é a motivação mais relevante. Isso pode ser explicado pelo fato de muitas equipes considerarem segurança como um item de qualidade de software, ou seja, quanto mais seguro o software for, maior a sua qualidade. A exigência do cliente por segurança no desenvolvimento de software alcançou 10,91% de motivação. Essa motivação está relacionada a determinado domínio que exige um nível mínimo de segurança como sistemas críticos, por exemplo. Melhoria na gestão de atividades de segurança e deficiência de segurança dos sistemas existentes obtiveram 1,82% e 0,91%, respectivamente.

Os porquês da implantação das técnicas e ferramentas estão apresentadas na Tabela 11.

Tabela 11 – Porquês das escolhas das técnicas e ferramentas para detecção de vulnerabilidades

Escolhas das técnicas e ferramentas	Quantidade %
Técnicas e ferramentas consolidadas no mercado	10,91
Aptidão e uso da ferramenta por parte da equipe	8,18
Escolha feita por um membro da equipe	5,45
Proporcionar qualidade do software	4,55
Exigência do cliente	4,55
Evitar problemas de segurança durante o projeto	4,55
Precisão da técnica e ferramenta na busca por vulnerabilidades	2,73
Escolha feita por terceiros a equipe	0,91
Não foram escolhidas técnicas e ferramentas para detecção de vulnerabilidades	20,91
Não sei	4,55
Branco	32,73
Total	100%

Fonte: Lígia Santos, 2017

As técnicas e ferramentas utilizadas pelos membros das equipes foram escolhidas de acordo com sua consolidação no mercado. Uma técnica e ferramenta pode ter boas referências no mercado por ser tendência, por ter bom custo-benefício, por ser recomendada para as soluções propostas ou ainda por ser a melhor opção disponível. Dessa forma, 10,91% dos respondentes alegaram esse porquê na escolha das técnicas e ferramentas para detecção de vulnerabilidades. Para 8,18% dos membros das equipes, a aptidão no uso das técnicas e ferramentas e necessidade de treinamento foi um motivo relevante para suas escolhas dentro dos projetos. É recomendável aos envolvidos no projeto que possuam um grau de aptidão mínimo nas técnicas e ferramentas para que a equipe esteja sincronizada em relação a suas implantações. O treinamento da equipe possibilita um melhor entendimento do funcionamento das técnicas e ferramentas. Um motivo da incorporação de uma técnica e ferramenta em um projeto é se ela é de fácil utilização.

A escolha de técnicas e ferramentas por parte de membros das equipes está relacionada a recomendações da própria organização ou sugestões da liderança da equipe. Assim, 5,45% dos respondentes alegaram esse porquê da escolha das técnicas e ferramentas.

Já para 4,55% dos respondentes, qualidade do software, exigência do cliente e diminuição de problemas com segurança nos projetos foram os porquês relatados. Muitas

equipes que adotam métodos ágeis como processo de desenvolvimento de software consideram segurança como um item de qualidade de software. A exigência do cliente é comum no porquê de escolha das técnicas e ferramentas, pois pode estar relacionada a exigências contratuais de alguns projetos. A diminuição de problemas futuros com segurança também motivou a escolha de determinadas técnicas e ferramentas nos projetos. A motivação da escolha relacionada a sua precisão foi pontuada por 2,73% dos membros das equipes. Caso a técnica ou ferramenta possua um bom desempenho na detecção de vulnerabilidades, ela pode ser implantada dentro de um projeto. Para 0,91% dos respondentes, a escolha foi definida por uma equipe terceirizada. Sendo assim, uma técnica e ferramenta consolidada no mercado é mais importante que utilizar uma que possua alto grau de precisão.

Os detalhes da implantação das técnicas e ferramentas estão apresentadas na Tabela 12.

Tabela 12 – Detalhes da implantação das técnicas e ferramentas para detecção de vulnerabilidades

Detalhes sobre a implantação das técnicas e ferramentas	Quantidade %
Pela própria equipe de forma incremental	10,00
Por meio de treinamento e difusão do conhecimento entre membros	9,09
Por indicação de membros da equipe	6,36
Necessidade do momento	4,55
Por meio de ferramentas e suas documentações	2,73
Inclusão de requisitos de segurança em projetos fechados	0,91
A implantação de segurança foi realizada por outra equipe	0,91
A implantação de segurança foi realizada em conjunto com o cliente	0,91
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	22,73
Não sei	6,36
Branco	35,45
Total	100%

Fonte: Lígia Santos, 2017

Para 10% dos respondentes, a implantação das técnicas e ferramentas ocorreu de forma incremental, ou seja, gradativamente até que toda equipe entendesse a importância da segurança nos projetos. Realização de pilotos, conversas, revisões da implantação das técnicas e ferramentas foram realizadas entre as equipes. Treinamento para difusão do conhecimento entre membros mais experientes por meio da composição de uma equipe experiente em segurança de software foi constatado entre os respondentes, com 9,09%. Para 6,36% dos respondentes, a implantação ocorreu por meio da indicação e coordenação

de algum membro da equipe. A necessidade do momento guiou a implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto. Ou seja, em algum projeto específico ou em determinada fase do desenvolvimento notou-se a necessidade da implantação de uma técnica e ferramenta. Sendo assim, 4,55% dos membros das equipes apontaram essa situação.

A implantação ocorreu por meio das técnicas e ferramentas e suas documentações para 2,73% dos respondentes. Elas auxiliavam na revisão do código e sua qualidade. Já para 0,91%, a implantação ocorreu por meio da inclusão de requisitos de segurança nos projetos; a atividade de segurança foi realizada por outra equipe; e implantação de segurança realizada juntamente ao cliente.

As estratégias da implantação das técnicas e ferramentas estão apresentadas na Tabela 13.

Tabela 13 – Estratégias utilizadas para a implantação dessas técnicas e ferramentas para detecção de vulnerabilidades

Estratégias utilizadas	Quantidade %
Por meio do uso de técnicas e ferramentas	10,91
Por meio de treinamento	6,36
De forma incremental	4,55
Por meio de metodologias existentes ou próprias	4,55
Consulta com especialistas e discussões para solucionar possíveis dúvidas	3,64
Por meio da organização da equipe e suas atividades	3,64
Implantação apenas nas áreas mais críticas	0,91
A implantação de segurança foi realizada por outra equipe	0,91
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	20,91
Não sei	6,36
Branco	37,27
Total	100%

Fonte: Lígia Santos, 2017

As estratégias apontadas pelos respondentes foram implementadas por meio do uso de técnicas e ferramentas para realização da análise estática e revisão de código. Também foram utilizadas técnicas de testes de revisão de segurança de código. Essa estratégia foi apontada por 10,91% dos respondentes. Treinamentos constantes com a equipe foi a estratégia definida por 6,36% dos participantes. Implantar segurança nos projetos de forma incremental foi a estratégia de 4,55% dos respondentes. Implantar segurança em

um projeto pode ser uma mudança drástica na rotina de algumas equipes. Dessa forma, é recomendável que a implantação ocorra de forma incremental, definindo pequenas tarefas e analisando os seus efeitos em outras áreas. Para 4,55% dos respondentes a estratégia de implantação utilizada foi baseada em metodologias próprias ou existentes.

A consulta com especialistas e discussões para solucionar possíveis dúvidas foi a estratégia adotada por 3,64% dos membros das equipes. Disseminar o conhecimento, organizar grupos de discussões e consultar especialistas dentro da própria equipe é uma forma de facilitar a implantação, pois a equipe pode ter uma maior motivação ao ver seus membros incentivando o uso de segurança durante o desenvolvimento de software.

Já a estratégia de implantação de segurança em áreas mais críticas foi adotada por 0,91% dos membros das equipes. Finalmente, a estratégia adotada de implantação por outra equipe obteve 0,91% das respostas. As dificuldades e limitações apontadas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 14.

Tabela 14 – Dificuldades e limitações encontradas na implantação de atividades de segurança

Dificuldades e limitações	Quantidade %
Falta de conhecimento dos envolvidos no projeto	12,73
Dificuldade na aplicação das técnicas e ferramentas para detecção de vulnerabilidades	7,27
Prazos das entregas	6,36
Falta de sincronia da equipe	5,45
Resistência da equipe ou gerência em adotar segurança no desenvolvimento de software	5,45
Alto investimento em segurança	1,82
Não há dificuldades na implantação	0,91
A implantação de segurança foi realizada por outra equipe	0,91
Problemas com correções de segurança após a entrega do software	0,91
Manter a compatibilidade com os sistemas legados	0,91
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	16,36
Não sei	4,55
Branco	36,36
Total	100%

Fonte: Lígia Santos, 2017

Ao serem questionados sobre as dificuldades e limitações encontradas na implantação das técnicas e ferramentas, 12,73% dos respondentes alegaram a falta de conhecimento dos envolvidos nos projetos. Diversos fatores podem explicar esse alto índice como: falta

de incentivo à equipe em realizar cursos, falta de treinamentos ou dificuldade na busca por materiais relacionados às técnicas e ferramentas. As dificuldades na aplicação das técnicas e ferramentas obtiveram 7,27%. Essa dificuldade está relacionada à dificuldade de entendimento dos materiais disponibilizados, escassez de fóruns de discussão sobre possíveis dúvidas de manuseio da ferramenta ou por problemas internos à equipe. Dificuldades e limitações relacionadas ao prazo das entregas obtiveram 6,36% das respostas. Por conta de os métodos ágeis exigirem entregas frequentes de funcionalidades, a pressão pela entrega dentro dos prazos e custos estipulados podem fazer com que a segurança não seja implantada ou seja implantada de forma incorreta.

A falta de sincronia da equipe foi uma dificuldade para 5,45% dos membros das equipes. Tal fator pode ser ocasionado pela falta de especialista em segurança na equipe, já que não existe um orientador de boas práticas de segurança, bem como um responsável pela definição de políticas de segurança a serem seguidas durante os projetos. Resistência da equipe é um problema para 5,45% dos membros das equipes. Isso ocorre devido ao entendimento de que ferramentas para detecção de vulnerabilidades devem ser implantadas apenas nas fases finais do desenvolvimento ou por considerar inconveniente a implantação de segurança, pois nunca ocorreu um ataque ao sistema. Já o alto investimento em segurança obteve 1,82% das respostas dos membros das equipes. As demais dificuldades e limitações obtiveram 0,91% das respostas. As lições aprendidas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 15.

Tabela 15 – Lições aprendidas com a implantação de técnicas e ferramentas para detecção de vulnerabilidades

Lições aprendidas durante a implantação	Quantidade %
Aumento da qualidade do software	9,09
Investimento em segurança e benefícios futuros	9,09
Treinamento e disseminação do conhecimento entre membros	7,27
Evitar retrabalho por questões de segurança	6,36
Melhoria na aplicação das técnicas e ferramentas	2,73
Proteção da informação	1,82
A implantação de segurança foi realizada por outra equipe	0,91
Possibilidade de aplicar segurança em métodos ágeis	0,91
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	20,00
Não sei	6,36
Branco	36,36
Total	100%

Fonte: Lígia Santos, 2017

Em relação às lições aprendidas, 9,09% dos respondentes alegaram que houve o aumento da qualidade do software desenvolvido pela equipe. Ainda 9,09% citaram que o investimento em segurança acarreta benefícios futuros. Esses benefícios estão relacionados à reutilização de componentes considerados seguros em outros projetos, à facilidade na correção de possíveis problemas e ao menor índice de ocorrências de problemas de segurança na aplicação. *Feedback* e treinamento entre membros da equipe é uma lição aprendida para 7,27% dos respondentes, pois difundir o conhecimento entre os membros, incentivar treinamentos e cursos é a maneira mais acessível para as equipes. Diminuição de retrabalho por questões de segurança, melhoria na aplicação das técnicas e ferramentas e proteção da informação obtiveram, respectivamente, 6,36%, 2,73% e 1,82% das respostas dos membros das equipes. Já para 0,91% dos respondentes, a lição aprendida foi a possibilidade de aplicar segurança em métodos ágeis.

7.4 Resultado das entrevistas com os gerentes

Neste capítulo são apresentados os resultados obtidos das entrevistas com 9 gerentes de equipes ágeis. Os dados foram interpretados por meio da técnica de análise de conteúdo e maiores detalhes podem ser consultados na Seção 5.3.4.

7.4.1 Características dos gerentes de equipes de desenvolvimento ágil de software

Na Tabela 16, são apresentadas as características, como função na equipe e anos na atual função, dos líderes participantes da entrevista. Foram constatadas as seguintes funções: gerentes de produtos, gerentes de sistemas, analista sênior, *scrum master* e *agile coach*. O cálculo é baseado na resposta dos 9 líderes participantes da entrevista.

Tabela 16 – Função dos respondentes e anos de atuação

		%
Função na equipe	<i>Agile coach</i>	44,44
	<i>Scrum master</i>	22,22
	Analista sênior	11,11
	Gerente de sistemas	11,11
	Gerente de produtos	11,11
	Total	100%
Anos na atual função	2 anos a 5 anos.	33,33
	Acima de 10 anos.	33,33
	0 a 1 ano.	22,22
	6 anos a 10 anos.	11,11
	Total	100%

Fonte: Lígia Santos, 2017

As funções de *agile coach* e *scrum master* são as que obtiveram maiores percentuais com, respectivamente, 44,44% e 22,22%. As demais funções como analistas sênior, gerente de sistemas e gerente de produto obtiveram 11,11%. Essa baixa porcentagem nas funções acima citadas podem ser explicadas pelo fato de os métodos ágeis dificilmente terem funções de gerência como ocorre no modelo tradicional de desenvolvimento de software, mas sim funções como *agile coaches*, *scrum masters* e líderes, as quais incentivam e direcionam a equipe na realização dos eventos e entrega de artefatos dos métodos ágeis.

Quanto aos anos na atual função, a maioria possui experiência entre 2 a 5 anos e acima de 10 anos com 33,33%. Já entre 0 a 1 ano de experiência na atual função obteve 22,22% e de 6 a 10 anos 11,11%.

7.4.2 Características dos projetos

Na Tabela 17, pode-se observar os métodos ágeis utilizados, tecnologias utilizadas, metodologias de implantação das técnicas e ferramentas, tamanho da equipe que atuou na implantação e responsáveis por segurança durante o desenvolvimento de software.

Uma mesma equipe pode desenvolver software com mais de um método ágil, atuar com diversas tecnologias e adotar mais de uma metodologia de implantação. Logo, o cálculo da porcentagem é baseado nas 9 respostas obtidas na entrevista.

Tabela 17 – Métodos ágeis utilizados, técnicas adotadas e composição da equipe e responsável pela implantação de segurança

		%
Métodos ágeis utilizados	Scrum + Kanban	55,56
	Scrum + Kanban + Extreme programming (XP)	22,22
	Kanban	11,11
	Diversos	11,11
	Total	100%
Tecnologia utilizada	Web	44,44
	Diversas	22,22
	Web + Mobile	11,11
	E-commerce	11,11
	R	11,11
	Total	100%
Metodologia de implantação	Metodologia própria	44,44
	Nenhuma	33,33
	Ferramentas	22,22
	Total	100%
Tamanho da equipe	De 0 a 5 pessoas.	44,44
	De 6 a 10 pessoas.	33,33
	Acima de 10 pessoas.	11,11
	Não sei.	11,11
	Total	100%
Responsável por segurança	Todos os membros da equipe	44,44
	Desenvolvedores	22,22
	Líderes	11,11
	Arquiteto de software	11,11
	Ninguém	11,11
	Total	100%

Fonte: Lígia Santos, 2017

Dessa forma, é possível observar que o *framework* ágil Scrum é utilizado conjuntamente com o método ágil Kanban, obtendo um percentual de 55,56%. Pelo fato de o Scrum ser um *framework*, ele tem a flexibilidade de ser utilizado conjuntamente com um método ágil específico. Ele pode ser utilizado para potencializar a eficiência e eficácia na aplicação das práticas ágeis, bem como adquirir maturidade o suficiente para a equipe seguir as práticas ágeis. Em relação ao uso de diversos métodos ágeis, é possível observar que 22,22% dos respondentes utilizam o Scrum, Kanban e Extreme Programming (XP). O contexto do projeto e necessidade explica a necessidade de se utilizar diferentes métodos ágeis e suas práticas. Uma prática ágil é diferente da outra, porém uma pode complementar ou potencializar a outra e isso de alguma forma traz benefícios para quem as utiliza.

O Kanban é um método que gerencia processos, seus fluxos e atividades. Sendo assim, obteve 11,11% do total de respostas. Diversos métodos ágeis obtiveram 11,11%.

Eles são ferramentas e práticas ágeis isoladas, independente de seguir um método ágil específico.

Em relação as tecnologias utilizadas pelos membros de equipes em seus projetos, a que obteve maior percentual é a *web* com 44,44%. Diversas tecnologias obtiveram 22,22%. Os respondentes apontaram que atuam com tecnologias como *web*, *mobile*, *e-commerce*, *desktop*, redes entre outras. Tecnologias *web* e *mobile* obtiveram 11,11% do percentual de respostas. *E-commerce* e tecnologias de manipulação de dados como o R obtiveram 11,11%.

As metodologias de implantação de segurança utilizadas pela maioria dos membros foram as próprias com 44,44%. Nenhuma metodologia obteve 33,33%. Ferramentas para gerenciar a implantação de técnicas e ferramentas obtiveram 22,22%.

Quanto ao tamanho da equipe, os membros apontaram que as equipes que implantaram as técnicas e ferramentas de segurança são compostas de até 5 pessoas, obtendo assim 44,44% das respostas. Entre 6 e 10 membros obtiveram 33,33% e acima de 10 membros ou não sabem obtiveram 11,11%.

Já em relação aos responsáveis pela implantação de segurança nos projetos, 44,44% dos membros apontaram que todos da equipe verificam tal aspecto, ou seja, é um trabalho compartilhado. 22,22% apontaram que os desenvolvedores são os responsáveis pela implantação de segurança nos projetos em que atuam. Para 11,11%, os líderes como coordenadores de TI, gerentes de projetos, gerentes de produtos, *scrum master* e *agile coach* são responsáveis pela implantação de segurança. A afirmação de que arquitetos de software são responsáveis por segurança foi apontada por 11,11% dos respondentes. Finalmente, não existe um responsável por essa atividade para 11,11%.

7.5 Técnicas e ferramentas para detecção de vulnerabilidades

A seguir, serão apresentadas as técnicas e ferramentas utilizadas em cada fase do ciclo de vida ágil pelos gerentes de equipes de desenvolvimento ágil de software.

7.5.1 Identificação e descrição das técnicas e ferramentas adotadas pelos gerentes de equipes ágeis

A Tabela 18 apresenta a relação das 18 técnicas e ferramentas de detecção de vulnerabilidades listadas de acordo com a revisão bibliográfica realizada. Elas foram classificadas nas seguintes fases: análise de requisitos e modelagem, *design*, codificação, testes e produção. A partir dessa classificação, foi possível analisar a porcentagem do uso atual de cada uma das técnicas e ferramentas, bem como sua utilização por fase do ciclo de vida de desenvolvimento de software. O cálculo está baseado nas 9 respostas obtidas na entrevista.

Tabela 18 – Uso atual de técnicas e ferramentas para detecção de vulnerabilidades nos projetos pelos gerentes

Fases	Técnicas e ferramentas	Sim %	Não %	Sim* %	Não* %
AR e mod.	Especialista em segurança	0,00	100,00	22,22	77,78
	Requisitos de segurança	0,00	100,00		
	<i>Abuse/Misuse Case/Stories</i>	22,22	77,78		
	Modelagem de ameaças	0,00	100,00		
	Modelagem de riscos	11,11	88,89		
	Ferramenta de modelagem de riscos e ameaças	0,00	100,00		
<i>Design</i>	<i>Design</i> de segurança	22,22	77,78	22,22	77,78
	Contramedidas de segurança	0,00	100,00		
	Avaliação de vulnerabilidades	22,22	77,78		
Codificação	Codificação segura	33,33	66,67	66,67	33,33
	Ferramenta de análise estática de código	22,22	77,78		
	Revisão de segurança de código	44,44	55,56		
	Ferramenta de revisão de código	33,33	66,67		
Testes	Ferramenta de análise dinâmica de código	11,11	88,89	33,33	66,67
	Teste de penetração	0,00	100,00		
	<i>Fuzz testing</i>	0,00	100,00		
	Teste baseado em riscos	22,22	77,78		
Prod.	Gestão de respostas a incidentes	33,33	66,67	33,33	66,67

* = Por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

Na visão dos gerentes, a fase de codificação possui maior grau de utilização de técnicas e ferramentas se comparadas às demais alcançando 66,67% de utilização. Esse alto grau de uso pode ser explicado pelo fato de as técnicas e ferramentas que compõem essa fase serem conhecidas e os membros terem maior familiaridade e interesse nelas. Um outro motivo é a facilidade na busca por documentação de ferramentas, materiais e treinamentos.

Dentre as técnicas e ferramentas que compõem essa fase, a revisão de segurança de código, codificação segura e ferramenta de segurança de código são as que possuem maiores graus de uso com, respectivamente, 44,44%, 33,33% e 33,33%.

As fases de testes e produção obtiveram 33,33% do grau de uso atual. Em relação à fase de testes, se comparado com a fase de codificação, é possível perceber que possui um percentual baixo. As técnicas de teste de penetração e *fuzz testing* não possuem grau de uso. Ferramenta de análise dinâmica e teste baseado em risco possuem, respectivamente, 11,11% e 22,22%. Sendo assim, é possível inferenciar que durante a fase de testes, as técnicas e ferramentas que a compõe não possui um grau de uso significativo, pois a segurança é fortemente avaliada e implantada na fase de codificação. Já em relação a fase de produção, a técnica de gestão de respostas a incidentes possui 33,33% de grau de uso.

Finalmente, as fases de análise de requisitos e *design* obtiveram 22,22%. Essas duas fases foram as que pontuaram menores graus de uso na visão dos gerentes. Na primeira, apenas *abuse/misuse cases/stories* e modelagem de riscos são utilizadas. Elas possuem os graus de uso de, respectivamente, 22,22% e 11,11%. As demais técnicas e ferramentas não são utilizadas. Já na segunda, as técnicas de *design* de segurança e avaliação de vulnerabilidades são utilizadas obtendo 22,22% e 22,22%. A técnica de contramedidas de segurança não é utilizada. O baixo percentual nas fases de análise de requisitos e *design* pode ser motivado pelo desconhecimento de técnicas e ferramentas específicas para segurança, pela possibilidade dos prazos estarem apertados, o custo de implantação de alguma técnica e ferramenta ou pela falta de membros capacitados para realização das atividades de segurança.

É de praxe no desenvolvimento tradicional de software realizar testes tardiamente, sejam eles para analisar a qualidade do software produzido, sejam para garantir a segurança. Porém, ao utilizar métodos ágeis, pelo fato de a entrega de funcionalidades ser de forma iterativa e incremental, é importante que haja um equilíbrio no uso das técnicas e ferramentas entre as fases. Dessa forma, durante todo o ciclo de vida ágil, o mínimo de segurança deve ser implantado, refinado e analisado, não deixando para as fases finais das entregas das funcionalidades a verificação de possíveis vulnerabilidades.

7.6 Aptidão, interesses e necessidades de treinamento

Nesta seção, serão tratadas as aptidões dos gerentes, interesses e necessidades de treinamento em técnicas e ferramentas para detecção de vulnerabilidades.

7.6.1 Aptidão dos gerentes de equipes ágeis em técnicas e ferramentas

A Tabela 19 apresenta as aptidões dos 9 gerentes que participaram da entrevista. Em relação ao grau de aptidão especialista, a fase de produção, composta pela técnica de gestão de respostas a incidentes, possui maior percentual de grau de aptidão com 22,22%. A fase de testes possui percentual de 2,78%. As demais fases, não obtiveram percentual para o grau de aptidão especialista. Dentre as técnicas e ferramentas da fase de testes, apenas *fuzz testing* obteve percentual de grau de aptidão de 11,11%. O alto grau de aptidão em gestão de respostas a incidentes pode ser explicado pelo fato de toda a equipe ser responsável pela segurança no projeto, como mostra a Tabela 17, cabendo ao gerente identificar e gerenciar possíveis incidentes de segurança.

Para o grau de aptidão básico, as fases que obtiveram maiores percentuais foram, respectivamente, codificação, testes e *design* com 19,44%, 16,67% e 14,81%.

Em relação a fase de codificação, dentre as técnicas que compõem tal fase, as que possuem percentual mais alto é a revisão de segurança de código, codificação segura e ferramenta de análise estática de código com, respectivamente, 44,44%, 22,22% e 11,11%. Ferramenta de revisão de código não obteve percentual de grau de aptidão.

Na fase de testes, a técnica com maior percentual de grau de aptidão básico é o *fuzz testing* com 33,33%. Todas as demais técnicas e ferramentas possuem 11,11%. Já a fase de *design*, contramedidas de segurança obteve 22,22% e as demais técnicas e ferramentas 11,11%.

Finalmente, em relação ao grau baixo, a fase em que os gerentes possuem o maior percentual de aptidão baixa de técnicas e ferramentas é a análise de requisitos e modelagem, *design* e testes com, respectivamente, 70,37%, 70,37% e 68,44%. Na primeira, as ferramentas de modelagem de ameaças e riscos e as técnicas de modelagem de ameaças e modelagem de riscos obtiveram maior percentual de baixo grau de aptidão com, respectivamente, 88,89% e 77,78%. Na fase de *design*, a técnica de *design* de segurança possui maior percentual de

baixo grau com 77,78%. Já na fase de testes, as técnicas de teste de penetração, análise dinâmica de código e teste baseado em riscos obtiveram, respectivamente, 88,89% e 66,67%. Em relação a fase de codificação, obteve também um alto percentual de baixo grau de aptidão com 52,78%. Já a fase de produção obteve 33,33% para baixo grau de aptidão.

Com esses percentuais é possível inferenciar que os gerentes possuem aptidão nas técnicas e ferramentas a nível básico e moderado. Poucos possuem nível alto de aptidão ou são considerados especialistas. Já a maioria não possui aptidão alguma nas técnicas e ferramentas para detecção de vulnerabilidades. Isso pode ser considerado reflexo da não utilização das técnicas e ferramentas nos projetos e pelo fato de a equipe toda ser responsável pela segurança, não sendo responsabilidade da gerência deter o maior grau de aptidão e conhecimento. Isso explica o fato da gerência possuir conhecimentos a nível básico e moderado. Porém, esse grau de aptidão da gerência poderia ser maior, pois o percentual ainda é baixo.

Tabela 19 – Aptidão dos gerentes em técnicas e ferramentas para detecção de vulnerabilidades

Fases	Técnicas e ferramentas	Bx. %	Bas. %	Mod. %	A. %	Esp. %	Bx.* %	Bas.* %	Mod.* %	A* %	Esp.* %
AR e mod.	Especialista em segurança	66,67	11,11	22,22	0,00	0,00	70,37	12,96	14,81	1,85	0,00
	Requisitos de segurança	44,44	33,33	11,11	11,11	0,00					
	<i>Abuse/Misuse Case/Stories</i>	66,67	11,11	22,22	0,00	0,00					
	Modelagem de ameaças	77,78	11,11	11,11	0,00	0,00					
	Modelagem de riscos	77,78	11,11	11,11	0,00	0,00					
	Ferramenta de modelagem de riscos e ameaças	88,89	0,00	11,11	0,00	0,00					
Design	<i>Design</i> de segurança	77,78	11,11	11,11	0,00	0,00	70,37	14,81	14,81	0,00	0,00
	Contramedidas de segurança	66,67	22,22	11,11	0,00	0,00					
	Avaliação de vulnerabilidades	66,67	11,11	22,22	0,00	0,00					
Codificação	Codificação segura	33,33	22,22	33,33	11,11	0,00	52,78	19,44	16,67	11,11	0,00
	Ferramenta de análise estática de código	77,78	11,11	0,00	11,11	0,00					
	Revisão de segurança de código	22,22	44,44	22,22	11,11	0,00					
	Ferramenta de revisão de código	77,78	0,00	11,11	11,11	0,00					
Testes	Ferramenta de análise dinâmica de código	66,67	11,11	11,11	11,11	0,00	69,44	16,67	8,33	11,11	2,78
	Teste de penetração	88,89	11,11	0,00	0,00	0,00					
	<i>Fuzz testing</i>	55,56	33,33	0,00	0,00	11,11					
	Teste baseado em riscos	66,67	11,11	22,22	0,00	0,00					
Prod.	Gestão de respostas a incidentes	33,33	11,11	33,33	0,00	22,22	33,33	11,11	33,33	0,00	22,22

A.R = Análise de Requisitos — Bx. = Baixa — Bas. = Básica — Mod. = Moderada — A = Alta — Esp. = Especialista — * = Média da aptidão por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

7.6.2 Interesse dos gerentes de equipes ágeis nas técnicas e ferramentas

A Tabela 20 apresenta a relação de técnicas e ferramentas nas quais os gerentes possuem interesse em receber treinamento.

Em relação ao grau muito interessado, a fase de *design* é a que possui maior percentual com 22,22%. Isso confirma a informação obtida por meio da Tabela 19 que aponta um percentual alto para o grau de aptidão baixo. Ou seja, é possível inferenciar que os gerentes possuem muito interesse em realizar treinamento nas técnicas e ferramentas para detecção de vulnerabilidades na fase de *design*. Porém, as técnicas e ferramentas de avaliação de vulnerabilidades e contramedidas de segurança são as que possuem maiores percentuais de grau de interesse com, respectivamente, 33,33% e 22,22%.

A fase de codificação possui um percentual de 19,44% em relação ao grau muito interessado. Isso confirma a informação obtida por meio da Tabela 19 que aponta um alto percentual para o grau de aptidão baixo na fase. Porém as técnicas de codificação segura e revisão de segurança de código obtiveram os maiores percentuais com grau muito interessado com, respectivamente, 33,33% e 22,22%.

A fase de testes possui um percentual de 11,11%. Dentre as técnicas que compõem essa fase, o *fuzz testing* possui o maior percentual com 22,22%. Mesmo sendo um percentual alto nessa fase, ele justifica o alto percentual de baixa aptidão conforme apresentado na Tabela 18.

Já as fases de produção e análise de requisitos e modelagem obtiveram os menores percentuais de muito interessado, mesmo tendo alta porcentagem de baixo grau de aptidão, o que indica que nem sempre os gerentes têm interesse em receber treinamento nas técnicas e ferramenta em que possuem baixo grau de aptidão.

Para o grau não interessado, as fases de testes, *design* e análise de requisitos e modelagem obtiveram os maiores percentuais com, respectivamente, 36,11%, 33,33% e 33,33%.

Na fase de testes, assim como na fase de análise de requisitos e modelagem as técnicas em que os gerentes possuem menor grau de interesse são o teste de penetração com 44,44% e a modelagem de ameaças com 44,44%. A falta de necessidade, falta de incentivo e custo de implantação pode ser um agente motivador do não interesse por parte dos gerentes nessas técnicas.

Já a fase de *design* todas as técnicas possuem o mesmo percentual para o grau não interessado obtendo 33,33%.

Tabela 20 – Interesse dos gerentes nas técnicas e ferramentas para detecção de vulnerabilidades

Fases	Técnicas e ferramentas	NI. %	LI. %	M. %	I. %	MI. %	NI.* %	LI.* %	M* %	I* %	MI.* %
AR e mod.	Especialista em segurança	33,33	44,44	11,11	11,11	0,00	33,33	27,78	22,22	14,81	1,85
	Requisitos de segurança	33,33	44,44	11,11	11,11	0,00					
	<i>Abuse/Misuse Case/Stories</i>	33,33	11,11	22,22	22,22	11,11					
	Modelagem de ameaças	44,44	33,33	22,22	0,00	0,00					
	Modelagem de riscos	22,22	22,22	33,33	22,22	0,00					
	Ferramenta de modelagem de riscos e ameaças	33,33	11,11	33,33	22,22	0,00					
Design	<i>Design</i> de segurança	33,33	22,22	11,11	22,22	11,11	33,33	29,63	3,70	11,11	22,22
	Contramedidas de segurança	33,33	33,33	0,00	11,11	22,22					
	Avaliação de vulnerabilidades	33,33	33,33	0,00	0,00	33,33					
Codificação	Codificação segura	33,33	22,22	0,00	11,11	33,33	27,78	22,22	2,78	27,78	19,44
	Ferramenta de análise estática de código	33,33	11,11	0,00	44,44	11,11					
	Revisão de segurança de código	11,11	22,22	0,00	44,44	22,22					
	Ferramenta de revisão de código	33,33	33,33	11,11	11,11	11,11					
Testes	Ferramenta de análise dinâmica de código	33,33	22,22	22,22	22,22	0,00	36,11	27,78	8,33	16,67	11,11
	Teste de penetração	44,44	33,33	0,00	11,11	11,11					
	<i>Fuzz testing</i>	33,33	33,33	0,00	11,11	22,22					
	Teste baseado em riscos	33,33	22,22	11,11	22,22	11,11					
Prod.	Gestão de respostas a incidentes	22,22	33,33	11,11	22,22	11,11	22,22	33,33	11,11	22,22	11,11

A.R = Análise de Requisitos — NI = Não interessado — LI = Levemente interessado — Mod. = Moderadamente interessado — I = Interessado — MI. = Muito interessado — * = Média de interesse por fase do ciclo de vida.

Fonte: Lígia Santos, 2017

A Tabela 21 agrupa as técnicas e ferramentas para detecção de vulnerabilidades, independente das fases do ciclo de vida de desenvolvimento de software. Esses conjuntos são compostos por técnicas e ferramentas das diferentes fases do ciclo de vida, o que pode contribuir para a ampliação do conhecimento na implantação de segurança de software durante todo o projeto.

O percentual de interesse em receber treinamento nesses conjuntos de técnicas e ferramentas é variado. Os conjuntos 6,4 e 2 são os que possuem maior percentual de interesse em receber treinamento com, respectivamente, 88,89%, 77,78% e 77,78%. O conjunto 6 é composto pelas técnicas de teste baseado em riscos e revisão de segurança de código. O conjunto 4 é composto por ferramenta de modelagem de riscos e ameaças, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.

Essa variação no percentual é motivada pela necessidade do contexto do projeto em que os gerentes estão atuando, por interesse pessoal em adquirir conhecimento nas técnicas e ferramentas que compõem essa fase, por exigência ou recomendação de segurança por parte dos clientes ou pela garantia da qualidade do software desenvolvido.

Porém, esse percentual pode variar por conta da baixa aptidão da equipe, problemas de custo e prazo ou pela falta de interesse em implantar técnicas e ferramentas para detecção de vulnerabilidades nos projetos.

Sendo assim, é benéfico ter interesse em receber treinamento em técnicas e ferramentas de diferentes fases do ciclo de vida de desenvolvimento de software, já que gerentes de equipes também devem gerenciar a segurança e incentivar, facilitar e difundir seus conhecimentos, mesmo que mínimos, em segurança de software, entre os membros das equipes.

Tabela 21 – Interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades

Interesse em receber treinamento	Sim %	Não %
1 - Especialista em segurança, requisitos de segurança e <i>abuse/misuse cases/stories</i> .	55,56	44,44
2 - Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	77,78	22,22
3 - <i>Design</i> de segurança, contramedidas de segurança e codificação segura.	44,44	55,56
4 - Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	77,78	22,22
5 - Avaliação de vulnerabilidades, teste de penetração e <i>Fuzz testing</i> .	44,44	55,56
6 - Teste baseado em riscos e revisão de segurança de código.	88,89	11,11

Fonte: Lígia Santos, 2017

7.6.3 Necessidade de treinamento dos gerentes de equipes ágeis nas técnicas e ferramentas

A Tabela 22 apresenta o grau de interesse dos gerentes em receber treinamento em um conjunto de técnicas e ferramentas. Sendo assim, para os graus “levemente interessado”, “muito interessado” e “não sei” obtiveram percentual de 0,00%.

Em relação ao grau moderado de interesse, os conjuntos 3 e 6 foram os que obtiveram maior percentual com, respectivamente, 44,44% e 33,33%. O conjunto 3 é composto pelas técnicas de *design* de segurança, contramedidas de segurança e codificação segura. Já o conjunto 6 é composto pelas técnicas de teste baseado em riscos e revisão de segurança de código. Para o grau não interessado, os maiores percentuais foram dos conjuntos 1, 2, 4 e 5, cada um com 66,67%.

Tabela 22 – Grau de interesse em receber treinamento em algum dos conjuntos de técnicas e ferramentas para detecção de vulnerabilidades

Grau de interesse em receber treinamento	NI. %	LI. %	Mod. %	MI. %	NS. %
1 - Especialista em segurança, requisitos de segurança e <i>abuse/misuse cases/stories</i> .	66,67	0,00	22,22	0,00	0,00
2 - Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	66,67	0,00	22,22	0,00	0,00
3 - <i>Design</i> de segurança, contramedidas de segurança e codificação segura.	44,44	0,00	44,44	0,00	0,00
4 - Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	66,67	0,00	22,22	0,00	0,00
5 - Avaliação de vulnerabilidades, teste de penetração e <i>Fuzz testing</i> .	66,67	0,00	22,22	0,00	0,00
6 - Teste baseado em riscos e revisão de segurança de código.	55,56	0,00	33,33	0,00	0,00

NI = Não interessado — LI = Levemente interessado — Mod. = Moderadamente interessado — MI = Muito interessado — NS. = Não sabe

Fonte: Lígia Santos, 2017

7.6.4 Características da implantação das técnicas e ferramentas

As características da implantação das técnicas e ferramentas para detecção de vulnerabilidades foram obtidas por meio de entrevista de 9 gerentes de equipes de desenvolvimento de software. As perguntas tratavam das motivações, dificuldades, limitações e lições aprendidas na sua implantação. Todas as perguntas tratadas nesta seção passaram pela técnica de análise de conteúdo conforme descrito na Seção 5.3.4. A Tabela 23 apresenta as motivações dos gerentes em relação à implantação de técnicas e ferramentas para detecção de vulnerabilidades no desenvolvimento ágil de software.

Tabela 23 – Motivação para a implantação de segurança no desenvolvimento ágil de software

Motivações da implantação	Quantidade %
Mitigação de riscos e ameaças	55,56
Qualidade do software	33,33
Exigência do cliente	11,11
Total	100%

Fonte: Lígia Santos, 2017

Ao serem questionados sobre as motivações da implantação de técnicas e ferramentas, 55,56% dos gerentes entrevistados alegaram a mitigação de riscos e ameaças como um agente principal para a implantação de segurança nos projetos. Já para 33,33% dos gerentes, a qualidade de software é a motivação mais relevante para implantar segurança. Em relação à mitigação de riscos e ameaças, é importante implantar alguma técnica e ferramenta para detecção de vulnerabilidades nos projetos, com o intuito de mitigar possíveis riscos e ameaças. Dessa forma, a equipe atuando conjuntamente com a garência deve modelar as respostas a possíveis riscos e ameaças na aplicação, auxiliando os membros das equipes na realização de outras atividades ao longo do ciclo de vida do desenvolvimento ágil de software. Já em relação à qualidade de software, a capacidade do software manter informações protegidas ou evitar acessos indevidos, por exemplo, para muitos gerentes, é considerado um aspecto de qualidade pelo fato de a segurança ser comumente tratada como um requisito, ou seja, se o software atende às necessidades dos clientes, ele é considerado de qualidade.

Os porquês da implantação das técnicas e ferramentas estão apresentados na Tabela

Tabela 24 – Escolha das técnicas e ferramentas para detecção de vulnerabilidades

Escolhas apontadas	Quantidade %
Melhor desempenho na busca por vulnerabilidades	33,33
Facilidade no entendimento e uso da técnica e ferramenta pela equipe	22,22
Recomendação ou exigência do cliente	22,22
Qualidade do software desenvolvido	11,11
Recomendação do arquiteto de software	11,11
Total	100%

Fonte: Lígia Santos, 2017

As técnicas e ferramentas utilizadas pelos gerentes foram escolhidas de acordo com o seu desempenho na busca por vulnerabilidades. Dessa forma, 33,33% dos gerentes alegaram esse porquê na escolha das técnicas e ferramentas a serem implantadas. A seleção das técnicas e ferramentas deve levar em conta o seu desempenho na busca por vulnerabilidades, caso contrário, a técnica e ferramenta selecionada será ineficaz.

A facilidade no entendimento e uso da técnica e ferramenta pela equipe foi apontado por 22,22% dos gerentes. Sendo assim, é recomendável que os membros das equipes e gerentes tenham algum domínio, confiança e se sintam confortáveis utilizando a técnica e ferramenta escolhida. Porém, nem todos membros de equipes possuem o mesmo grau de aptidão. Sendo assim, cursos e treinamentos devem ser promovidos, bem como a difusão do conhecimento entre membros experientes e demais membros.

Recomendação ou exigência do cliente foi um dos aspectos mais relevantes tratados por 22,22% dos gerentes. Sendo assim, os gerentes e membros de equipes devem atender as especificações de segurança listadas pelos clientes. Eles exigem segurança em seus projetos, listando aspectos que devem ser seguidos rigorosamente. Isso é motivado pelo fato de os clientes lidarem com dados sensíveis, exigindo que a aplicação seja confiável.

A qualidade de software foi tratada por 11,11% dos gerentes entrevistados. Sendo assim, uma técnica e ferramenta é implantada para proporcionar qualidade ao software desenvolvido. Uma ferramenta citada pelos gerentes na entrevista foi a Sonar (SONAR-SOURCE, 2017). Essa ferramenta avalia a qualidade do código, apontando o seu grau de complexidade e ocorrência de mal cheiro de código (FOWLER; BECK, 1999), por exemplo. Já para 11,11%, a escolha da técnica e ferramenta ocorreu por meio da recomendação do arquiteto de software. Essa recomendação pode ter sido motivada por diversos fatores,

como a facilidade na implantação, facilidade no uso, melhor custo benefício e melhor desempenho na busca por vulnerabilidades.

Os detalhes da implantação das técnicas e ferramentas estão apresentadas na Tabela 25.

Tabela 25 – Detalhes sobre a implantação das técnicas e ferramentas para detecção de vulnerabilidades

Detalhes apontados	Quantidade %
Por meio de ferramentas e suas documentações	55,56
Não estava no momento da implantação ou não sabe	22,22
Pela própria equipe, por meio de treinamento	11,11
Comparação das técnicas e ferramentas utilizadas no mercado	11,11
Total	100%

Fonte: Lígia Santos, 2017

Para 55,56% dos respondentes, a implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto ocorreu por meio da sua documentação. Ela serviu como um guia, contribuindo para um melhor entendimento do funcionamento da técnica e ferramenta por parte da gerência e demais membros. 22,22% dos respondentes alegaram não estar no momento da implantação das técnicas e ferramentas ou não sabiam detalhes da implantação.

A implantação foi realizada pela própria equipe por meio de treinamento para 11,11% dos gerentes. Logo, foram realizados treinamentos nas técnicas e ferramentas que seriam utilizadas ao longo do projeto e após isso realizaram a implantação. Já para 11,11% dos gerentes, a implantação ocorreu por meio das técnicas e ferramentas mais utilizadas no mercado. Sendo assim, foi realizada uma busca por técnicas e ferramentas de segurança e selecionadas as mais adequadas aos projetos, que possuíam maiores custo benefício e que tinham melhor desempenho na detecção de vulnerabilidades. As estratégias da implantação das técnicas e ferramentas estão apresentadas na Tabela 26.

Tabela 26 – Estratégias utilizadas para a implantação das técnicas e ferramentas para detecção de vulnerabilidades

Estratégias utilizadas	Quantidade %
Estratégias utilizando técnicas e ferramentas existentes	66,67
Colaboração entre membros da equipe	33,33
Total	100%

Fonte: Lígia Santos, 2017

As estratégias apontadas durante a entrevista foram implementadas por meio do uso de técnicas e ferramentas existentes. Dessa forma, 66,67% dos gerentes usaram as técnicas e ferramentas existentes de acordo com a necessidade do projetos. Já para 33,33% dos gerentes, a colaboração entre membros foi a estratégia adotada. Sendo assim, essa estratégia contribuiu para a difusão do conhecimento entre membros, intensificando o trabalho em equipe, proporcionando *feedback* constante, aumentando o conhecimento, autonomia e segurança na implantação das técnicas e ferramentas para detecção de vulnerabilidades. A contribuição vinda de membros experientes é eficaz na difusão de conhecimento e incentivo do uso das técnicas e ferramentas. As dificuldades e limitações apontadas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 27

Tabela 27 – Dificuldades e limitações encontradas na implantação

Dificuldades e limitações apontadas	Quantidade %
Necessidade de mudança de pensamento do pessoal	33,33
Falta de conhecimento dos membros da equipe	22,22
Cultura organizacional não sensibilizada para segurança	11,11
Problemas de orçamento do projeto	11,11
Dificuldade na busca por informações de segurança	11,11
Não menciona	11,11
Total	100%

Fonte: Lígia Santos, 2017

A maioria das dificuldades e limitações ocorridas durante a implantação das técnicas e ferramentas para detecção de vulnerabilidades foram ocasionadas pela resistência da equipe em relação ao novo. Sendo assim, é necessário a mudança do pensamento, incentivando o uso das técnicas e ferramentas, apresentando seus benefícios e o que a ausência de segurança proporciona à aplicação. Essas dificuldades e limitações foram apontadas por 33,33% dos gerentes entrevistados.

A falta de conhecimento dos membros da equipe foi um dos fatores que dificultou e limitou a implantação. Essa ocorrência pode ser explicada pelo fato de a equipe ter desconhecimento nas técnicas e ferramentas selecionadas para a implantação, a falta de treinamento e a dificuldade no seu manuseio. Dessa forma, essa dificuldade e limitação foi tratada por 22,22% dos gerentes entrevistados.

A cultura organizacional não sensibilizada para segurança foi um fator que dificultou e limitou a implantação para 11,11% dos gerentes. A organização deve incentivar as equipes a realizarem a implantação de segurança, utilizando técnicas e ferramentas para detecção de vulnerabilidades adequadas em seus projetos. Se tal aspecto não parte da organização, a gerência e os demais membros terão dificuldades em introduzir e manter a cultura de implantação de segurança nos projetos.

Problemas com o orçamento do projeto foi um fator que dificultou e limitou a implantação para 11,11% dos gerentes entrevistados. Uma técnica e ferramenta pode possuir um melhor desempenho em um projeto específico, sendo assim, caso ela tenha um custo alto de aquisição, poderá aumentar significativamente o custo dos projetos.

As dificuldades na busca por informações sobre segurança como as técnicas e ferramentas que serão implantadas e vulnerabilidades existentes foi um fator que dificultou e limitou a implantação para 11,11% dos gerentes entrevistados. Algumas técnicas e ferramentas possuem documentações que não apresentam suas informações de forma clara ou didática, dificultando o seu uso. Além disso, a equipe desconhece os materiais que tratam de possíveis vulnerabilidades nas aplicações, suas causas e possíveis mitigações. Já 11,11% dos entrevistados não mencionaram as dificuldades e limitações ocorridas durante a implantação.

As lições aprendidas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 28.

Tabela 28 – Lições aprendidas com a implantação das técnicas e ferramentas para detecção de vulnerabilidades

Lições aprendidas	Quantidade %
Necessidade em haver trabalho em equipe e troca de conhecimento	33,33
Relevância da segurança de software em um projeto	33,33
Utilizar ferramentas de segurança	22,22
Não tem	11,11
Total	100%

Fonte: Lígia Santos, 2017

Para 33,33% dos gerentes entrevistados, as lições aprendidas foram em relação ao trabalho em equipe, cooperação entre membros e troca de conhecimento. Um dos quatro valores dos métodos ágeis está relacionado aos indivíduos e interações, mais que processos e ferramentas. Dessa forma o trabalho em equipe e interações entre membros a fim de auxiliar uns aos outros em relação a implantação é recomendável. O *feedback*, interação, colaboração e incentivo entre gerência e demais membros da equipe no uso de técnicas e ferramentas para detecção de vulnerabilidades é o agente principal que possibilita bons resultados na implantação. A técnica e ferramenta traz bons resultados, mas o que de fato garante que a implantação ocorreu de forma correta, dentro dos valores e princípios ágeis, é o *feedback*, interação, colaboração e o incentivo entre membros, a fim de melhorar continuamente as próximas implantações.

A importância da segurança em projetos foi uma das lições aprendidas tratada por 33,33% dos gerentes. Negligenciar segurança nos projetos é comum, visto que muitos consideram a segurança um aspecto relacionado à qualidade de software. É necessário apresentar os benefícios da implantação de segurança, bem como os impactos de sua ausência. Sendo assim, é recomendável que seja dada a devida atenção à segurança de software, pois a sua ausência pode causar prejuízos às organizações.

A utilização de técnicas e ferramentas para detecção de vulnerabilidades nos projetos foi uma das lições aprendidas para 22,22% dos gerentes entrevistados. A sua implantação traz benefícios e sua ausência prejuízos. Dessa forma, é importante avaliar se as técnicas e ferramentas candidatas à implantação são as adequadas dentro dos pontos previamente definidos, bem como sua compatibilidade com o projeto.

Já 11,11% dos gerentes entrevistados não apontaram lições aprendidas durante a implantação das técnicas e ferramentas para detecção de vulnerabilidades.

7.7 Ameaças à validade

Os resultados obtidos por meio de um projeto de pesquisa são influenciados por fatores que afetam a sua validade externa, interna e de construto.

A validade externa indica a capacidade de os resultados de um projeto de pesquisa serem estendidos a outros. Os resultados referentes ao uso de técnicas e ferramentas para detecção de vulnerabilidades estão limitados aos 18 encontrados na literatura. Essas técnicas e ferramentas compõem três dos principais processos para desenvolvimento seguro de software, os quais foram encontrados na literatura, são eles: Processo de McGraw (POTTER; MCGRAW, 2004), OWASP CLASP (GREGOIRE et al., 2007) e as atividades de Howard e Lipner (HOWARD; LIPNER, 2006). Essas técnicas e ferramentas foram selecionadas por serem comum a esses três processos. Logo, os resultados obtidos estão limitados a esses processos. Todavia, existem outros processos de desenvolvimento seguro de software compostos por técnicas e ferramentas para detecção de vulnerabilidades, as quais não foram avaliadas neste projeto de pesquisa.

A validade interna avalia as condições básicas aplicáveis ao estudo. Sendo assim, neste trabalho, a validade interna é influenciada pela quantidade de participantes no experimento. Foram encontrados poucos membros de equipes que implantaram técnicas e ferramentas para detecção de vulnerabilidades em projetos que aplicam métodos ágeis. A maioria dos participantes as implanta para promover a qualidade da aplicação desenvolvida.

Assim, a maior dificuldade foi a busca por membros de equipes que implantaram alguma das 18 técnicas e ferramentas de segurança no projeto. Houve também a tentativa de aplicar o *survey* em diferentes organizações e suas equipes de desenvolvimento, porém por questões burocráticas foi inviabilizada essa abordagem. A busca por uma amostra significativa de profissionais atuantes na indústria, que tivessem experiência em implantação de segurança foi difícil, e esse limitante motivou a ampliação do escopo. Como consequência disso, a condução do *survey* ocorreu com profissionais que atuassem em equipes de desenvolvimento de software que implantaram, ainda não implantaram ou implantaram parcialmente alguma técnica ou ferramenta para detecção de vulnerabilidades conjuntamente aos métodos ágeis. Sendo assim, os 110 participantes foram contatados por meio da rede de profissionais LinkedIn, por ela ser conceituada e muito utilizada pelas organizações e profissionais. Ou seja, o *survey* foi limitado a apenas essa rede, deixando

de lado outras formas de contato, as quais possivelmente teriam participantes que se encaixariam ao perfil desta pesquisa. Pelo fato de alguns membros de equipes atuarem em empresas de pequeno porte, alguns desses profissionais acabam acumulando funções na equipe. Os resultados obtidos nos experimentos poderiam ter sido diferentes se o *survey* fosse limitado apenas a membros que implantaram de forma total as técnicas e ferramentas. Porém, isso limitaria o escopo e dificilmente seria possível alcançar a participação de 110 membros de equipes.

Em relação ao questionário, alguns dos participantes não responderam de forma integral, ou ainda, não implantaram técnicas e ferramentas para detecção de vulnerabilidades, mas possuem uma intenção, motivação, ou um porquê para sua implantação, por exemplo. Assim, pode ser que exista alguma interferência dessas respostas em outras. Os resultados obtidos foram manipulados por meio das técnicas de estatística descritiva e análise de conteúdo. Outras abordagens poderiam ser aplicáveis ao projeto de pesquisa. Porém, pela dificuldade na busca por participantes que se encaixassem no escopo inicial deste projeto, optou-se por esse desenho experimental.

7.7.1 Considerações finais

Este capítulo apresentou os resultados do *survey* aplicado aos membros e das entrevistas conduzidas com os gerentes de equipes ágeis. O próximo capítulo apresenta as considerações finais desta dissertação, a síntese do trabalho, as possíveis contribuições e os trabalhos futuros.

8 Conclusões

A seção a seguir apresenta a síntese desta dissertação, as contribuições e os trabalhos futuros.

8.1 Síntese do trabalho

Esta dissertação teve como objetivo identificar e descrever o estado da prática da utilização de técnicas e ferramentas para detecção de vulnerabilidades durante o desenvolvimento ágil de software.

Para tanto, os seguintes objetivos específicos foram definidos: (1) descrever as características dos membros de equipes de desenvolvimento ágil de software; (2) descrever as características dos projetos em que atuam; (3) identificar as técnicas e ferramentas para detecção de vulnerabilidades utilizadas pelos membros de equipes ágeis; (4) identificar o grau de implantação das técnicas e ferramentas e os benefícios obtidos; (5) identificar as aptidões, os interesses e as necessidades de treinamento dos membros de equipes nas técnicas e ferramentas; e (6) identificar as motivações, as dificuldades, as limitações e as lições aprendidas durante a implantação.

Os objetivos específicos foram alcançados por meio de um *survey* respondido por 110 membros de equipes de desenvolvimento ágil de software. A coleta de dados foi realizada por meio de entrevistas e questionários aplicados a gerentes e demais membros das equipes ágeis para obter duas visões diferentes: a visão da equipe e a visão da gerência ou liderança.

Os resultados do *survey* permitiram avaliar o estado da prática do uso de ferramentas de técnicas e ferramentas para detecção de vulnerabilidades em equipes ágeis. Eles indicam que há uma parcela significativa de membros de equipes ágeis que relatam não saber ou que não responderam quais as motivações, as estratégias, as dificuldades, as limitações ou as lições aprendidas no uso dessas técnicas e ferramentas.

Dentre aqueles membros de equipes ágeis que responderam o questionário foi encontrado um maior uso nas técnicas e ferramentas que compõem a fase de codificação. Porém, o uso individual dessas técnicas e ferramentas não alcança 50%. A técnica de codificação segura é a que obteve maior percentual. Já em relação aos gerentes, o uso atual na fase de codificação obteve 66,67%. A técnica de revisão de segurança de código foi a que obteve maior percentual com 44,44%.

Em relação à aptidão, a fase de testes foi a que os membros de equipes alegaram possuir menor habilidade. As técnicas de *fuzz testing* e de teste baseado em riscos são as mais apontadas. Já os gerentes alegaram possuir baixa aptidão nas fases de análise de requisitos e modelagem e *design*. As técnicas e ferramentas em que eles possuem menos aptidão são as ferramentas de modelagem de riscos e ameaças e *design* de segurança.

Em relação ao grau de interesse, os membros estão mais interessados em técnicas e ferramentas da fase de codificação, especialmente na codificação segura, que é a mais apontada. Já os gerentes possuem mais interesse nas fases de *design* e codificação, especialmente nas técnicas de avaliação de vulnerabilidades e codificação segura.

Em relação ao grau de implantação e benefícios, o maior grau deu-se na fase de codificação. A técnica de codificação segura e ferramentas de análise estática de código foram as que possuíram mais pontos de implantação e benefícios associados ao seu uso. Já as técnicas de *fuzz testing* e teste de penetração são as menos implantadas e que possuem menos ocorrências de benefícios associados.

Em relação às características da implantação, os membros de equipes ágeis e os gerentes identificaram a mitigação de riscos e ameaças como a maior motivação da implantação de segurança. Em relação ao porquê da implantação, os membros apontaram a consolidação da técnica no mercado; já os gerentes apontaram o melhor desempenho na busca por vulnerabilidades. Em relação à estratégia adotada, os membros apontaram que a implantação ocorreu de forma incremental; já os gerentes apontaram apenas o uso das técnicas e ferramentas e suas documentações.

As dificuldades apontadas pelos gerentes estão relacionadas à mudança de pensamento da equipe; já os membros citaram a falta de conhecimento dos envolvidos no projeto. Finalmente, quanto às lições aprendidas, os gerentes apontaram a necessidade de trabalho em equipe e troca de conhecimento, bem como a relevância da segurança de software em um projeto. Os membros apontaram o aumento da qualidade do software, os benefícios futuros e o treinamento e disseminação do conhecimento entre membros.

Os dados obtidos também indicam que atividades de segurança são ainda pouco utilizadas por esses profissionais. Possíveis ações como a colaboração entre membros, o compartilhamento de informação e a auto-organização da equipe podem fazer com que as equipes ágeis adotem corretamente as técnicas e ferramentas. A sua baixa utilização pode fazer com que os membros desenvolvam uma baixa aptidão, pouco interesse ou resistência.

Uma limitação do trabalho são o alto número de respostas em branco e que não implantaram detecção de vulnerabilidades. Isso reflete a falta de oportunidade ou de preocupação em conciliar segurança e métodos ágeis. Muitos dos membros apontaram ter motivações, porém, existem dificuldades para realizar a implantação; elas geralmente estão relacionadas à falta de cultura organizacional voltada para segurança, aos prazos, ao desconhecimento das técnicas e ferramentas, à resistência da equipe e ao custo da implantação.

8.2 Possíveis contribuições

Esta dissertação apresentou um *survey* que identificou, descreveu e comparou o estado da prática na utilização de técnicas e ferramentas para segurança pelos membros de equipes ágeis. Seus resultados podem servir de base para profissionais que visam implantar segurança em seus projetos ágeis. Eles descrevem as características dos membros das equipes ágeis e de seus projetos, o seu nível de aptidão, de interesses e de necessidade de treinamento. Além disso, o grau de implantação das técnicas e ferramentas e os benefícios obtidos indicam o impacto das atividades de segurança nos produtos de software.

8.3 Trabalhos futuros

São ainda poucos os estudos que combinam métodos ágeis e desenvolvimento de software seguro, avaliando o estado da prática dos membros das equipes, as suas aptidões, as suas necessidades de treinamento e seus interesses. Além disso, poucos tratam dos benefícios obtidos com a implantação e quais as técnicas e ferramentas que podem ser incorporadas em cada uma das fases do ciclo ágil. Assim, os seguintes trabalhos futuros são propostos para dar prosseguimento a esta pesquisa.

- Identificação e comparação de outros processos de desenvolvimento de software seguro.
- Inclusão de novas técnicas e ferramentas que compõem esses processos, permitindo que elas sejam incluídas nas fases do ciclo de vida ágil.
- Proposta de um modelo para desenvolvimento de software seguro, considerando os processos, as práticas, as atividades ágeis e a segurança.

- Avaliação do estado da prática do uso das técnicas e ferramentas “in loco”, isto é, nas organizações.
- Aplicação do *survey* em equipes completas e não com membros de distintas equipes.
- Avaliação do uso de técnicas e ferramentas em fases específicas do ciclo de vida ágil.
- Uso de outras técnicas estatísticas para identificar novas ocorrências e confirmar as afirmações já obtidas neste estudo.

Referências¹

- AHO, A.; SETHI, R.; ULLMAN, J. *Compilers: principles, techniques, & tools*. 2^a. ed. [S.l.]: Addison-Wesley Publishing Company, Boston, MA, USA, 2007. 687–700 p. Citado 2 vezes nas páginas 45 e 46.
- ALEXANDER, I. Misuse cases: use cases with hostile intent. *IEEE software*, IEEE, v. 20, n. 1, p. 58–66, 2003. Citado na página 39.
- AMBLER, S. *Agile modeling: effective practices for extreme programming and the unified process*. [S.l.]: John Wiley & Sons, 2002. Citado na página 27.
- AMBLER, S. W. *The agile unified process (AUP)*. 2006. Disponível em: <http://www.ambysoft.com/unifiedprocess/agileUP.html>. Citado 2 vezes nas páginas 27 e 80.
- BACA, D. et al. A novel security-enhanced agile software development process applied in an industrial setting. In: IEEE. *Availability, Reliability and Security (ARES), 2015 10th International Conference on*. [S.l.], 2015. p. 11–19. Citado 2 vezes nas páginas 61 e 63.
- BACA, D.; CARLSSON, B. Agile development with security engineering activities. In: ACM. *Proceedings of the 2011 International Conference on Software and Systems Process*. [S.l.], 2011. p. 149–158. Citado 3 vezes nas páginas 61, 62 e 65.
- BACH, J. Heuristic risk-based testing. *Software testing and quality engineering magazine*, v. 11, n. 9, 1999. Citado na página 48.
- BARDIN, L. *Análisis de contenido*. [S.l.]: Ediciones Akal, 1991. v. 89. Citado 2 vezes nas páginas 74 e 75.
- BARTSCH, S. Practitioners’ perspectives on security in agile development. In: IEEE. *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. [S.l.], 2011. p. 479–484. Citado 2 vezes nas páginas 61 e 62.
- BAUMEISTER, H.; WIRSING, M. Applying test-first programming and iterative development in building an e-business application. In: *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*. [S.l.: s.n.], 2002. p. 1–6. Citado na página 57.
- BECK, K. *Extreme programming explained*. 2^a. ed. [S.l.]: Addison-Wesley, 2000. 37–51 p. Citado 4 vezes nas páginas 26, 27, 28 e 80.
- BECK, K.; BEEDLE, M. et al. *Manifesto ágil de software*. 2001. Disponível em: <http://agilemanifesto.org/>. Citado 5 vezes nas páginas 18, 19, 23, 24 e 25.
- BEZNOSOV, K.; KRUCHTEN, P. Towards agile security assurance. In: ACM. *Proceedings of the 2004 workshop on New security paradigms*. [S.l.], 2004. p. 47–54. Citado na página 61.
- BLAIR, J.; CZAJA, R. F.; BLAIR, E. A. *Designing surveys: a guide to decisions and procedures*. [S.l.]: Sage Publications, 2013. Citado 2 vezes nas páginas 67 e 68.

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

- BSIMM. *About BSIMM - Bringing science to software security*. 2016. Disponível em: <https://www.bsimm.com/about/>. Citado 2 vezes nas páginas 54 e 55.
- CCMB. *Common Criteria for Information Technology Security Evaluation*. 2012. Disponível em: <https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf>. Citado na página 65.
- CERT.BR. *Cartilha de segurança para internet - CERT.br: Ataques na internet*. 2012. Disponível em: <http://cartilha.cert.br/ataques/>. Citado 2 vezes nas páginas 35 e 38.
- CHIEN, E.; SZÖR, P. Blended attacks exploits, vulnerabilities and buffer overflow techniques in computer viruses. *Virus Bulletin Conference, Virus Bulletin Ltd, The Pentagon, Abingdon, Oxfordshire, OX14 3YP, England.*, Symantec, v. 1, p. 1–14, 2002. Citado na página 38.
- CICHONSKI, P. et al. Computer security incident handling guide. *NIST Special Publication*, v. 800, p. 61, 2012. Citado na página 49.
- CIGITAL. *Penetration testing*. 2016. Disponível em: <https://www.cigital.com/services/penetration-testing/>. Citado na página 48.
- COCKBURN, A. *Crystal clear a human-powered methodology for small teams*. [S.l.]: Addison-Wesley Professional, 2004. Citado 2 vezes nas páginas 27 e 80.
- CRUZES, D. S. et al. “How is security testing done in agile teams? a cross-case analysis of four software teams”. In: *International Conference on Agile Software Development*. [S.l.]: Springer, 2017. p. 201–216. Citado 2 vezes nas páginas 60 e 61.
- CWE, C. W. E. *Common weakness enumeration: Heap-based buffer overflow*. 2015. Disponível em: <https://cwe.mitre.org/data/definitions/122.html>. Citado na página 38.
- DAVIES, R. *DSDM Explained*. 2004. Disponível em: <http://www.agilexp.com/presentations/DSDMexplained.pdf>. Citado na página 81.
- FOSTER, J. S.; FÄHNDRIK, M.; AIKEN, A. A theory of type qualifiers. *ACM SIGPLAN Notices*, ACM, v. 34, n. 5, p. 192–203, 1999. Citado na página 46.
- FOWLER, M.; BECK, K. *Refactoring: Improving the design of existing code*. 1ª. ed. [S.l.]: Addison-Wesley, USA, 1999. 75–87 p. Citado 2 vezes nas páginas 73 e 120.
- FRANQUEIRA, V. N. et al. Towards agile security risk management in re and beyond. In: IEEE. *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on*. [S.l.], 2011. p. 33–36. Citado 2 vezes nas páginas 61 e 64.
- FREITAS, M. A. dos S. *Fundamentos do gerenciamento de serviços De TI: preparatório para a certificação ITIL V3 foundation*. [S.l.]: Brasport, 2013. v. 2. Citado 2 vezes nas páginas 81 e 158.
- FREITEZ, W. R. J.; MAMMAR, A.; CAVALLI, A. R. Software vulnerabilities, prevention and detection methods: a review. In: *SEC-MDA: Security in Model Driven Architecture*. [S.l.: s.n.], 2009. p. 1–11. Citado 2 vezes nas páginas 45 e 48.

GREGOIRE, J. et al. On the secure software development process: CLASP and SDL compared. In: IEEE COMPUTER SOCIETY. *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*. [S.l.], 2007. p. 1–7. Citado 4 vezes nas páginas 53, 61, 64 e 125.

GRIESKAMP, W. et al. Generating finite state machines. *ACM SIGSOFT*, ACM, p. 112–122, 2002. Citado na página 35.

GROUP, I. 1044-2009-IEEE Standard classification for software anomalies. *IEEE, New York*, IEEE, 2010. Citado 2 vezes nas páginas 35 e 36.

HOWARD, M.; LIPNER, S. *The security development lifecycle*. [S.l.]: Microsoft Press, 2006. 225—239 p. Citado 9 vezes nas páginas 18, 20, 34, 35, 44, 51, 56, 57 e 125.

ISAAC, S.; MICHAEL, W. B. *Handbook in research and evaluation: A collection of principles, methods, and strategies useful in the planning, design, and evaluation of studies in education and the behavioral sciences*. [S.l.]: Edits publishers, 1995. Citado na página 67.

ISO. *ISO/IEC 21827:2008 Information technology – Security techniques – Systems Security Engineering – Capability Maturity Model® (SSE-CMM®)*. 2008. Disponível em: <https://www.iso.org/standard/44716.html>. Citado na página 65.

ISTQB. *Advanced level syllabus test manager*. 2012. Disponível em: <https://www.istqb.org/54-advanced-level-syllabus-2012-test-manager>. Citado na página 48.

JAATUN, M. G. Hunting for aardvarks: can software security be measured? In: SPRINGER. *Department of Software Engineering, Safety and Security. International Conference on Availability, Reliability, and Security*. [S.l.], 2012. p. 85–92. Citado na página 55.

JACOBSON, I. *Software development team essentials*. 2016. Disponível em: <https://www.ivarjacobson.com/software-development-team-essentials>. Citado na página 24.

JONES, C. Software defect origins. International Function Point Users Group, p. 1–31, 2013. Citado na página 36.

KERAMATI, H.; MIRIAN-HOSSEINABADI, S.-H. Integrating software development security activities with agile methodologies. In: IEEE. *IEEE/ACS International Conference on Computer Systems and Applications*. [S.l.], 2008. p. 749–754. Citado 3 vezes nas páginas 56, 61 e 64.

KHAN, M. U. A.; ZULKERNINE, M. A survey on requirements and design. *Queen's University*, 2009. Citado 6 vezes nas páginas 39, 43, 51, 52, 53 e 54.

KIM, H. et al. Applying dataflow analysis to detecting software vulnerability. In: IEEE. *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*. [S.l.], 2008. v. 1, p. 255–258. Citado na página 46.

LARMAN, C. *Applying UML and patterns: an introduction to object oriented analysis and design and interative development*. [S.l.]: Pearson Education India, 2012. Citado na página 27.

LEFFINGWELL, D. *Scaling software agility: best practices for large enterprises*. 1^a. ed. [S.l.]: Pearson Education, Boston, MA, 2007. 1–94 p. Citado na página 32.

LINDVALL, M. et al. Agile software development in large organizations. *IEEE Computer*, IEEE, v. 37, n. 12, p. 26–34, 2004. Citado na página 33.

LIPNER, S.; HOWARD, M. *O ciclo de vida do desenvolvimento da segurança de computação confiável*. 2005. Disponível em: <https://msdn.microsoft.com/pt-br/library/ms995349.aspx>. Citado na página 43.

MÁRQUEZ, G. et al. Identifying emerging security concepts using software artifacts through an experimental case. In: IEEE. *Chilean Computer Science Society (SCCC), 2015 34th International Conference of the*. [S.l.], 2015. p. 1–6. Citado 2 vezes nas páginas 61 e 62.

MCDERMOTT, J.; FOX, C. Using abuse case models for security requirements analysis. In: IEEE. *Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual*. [S.l.], 1999. p. 55–64. Citado na página 40.

MCGRAW, G.; MORRISETT, G. Attacking malicious code. *IEEE*, IEEE Computer Society, p. 1–11, 2000. Citado na página 36.

MICROSOFT. *Type qualifiers*. 2015. Disponível em: <https://msdn.microsoft.com/en-us/library/888bfst6.aspx>. Citado na página 46.

MICROSOFT. *What is the security development lifecycle?* 2016. Disponível em: <https://www.microsoft.com/en-us/sdl/>. Citado na página 52.

MICROSOFT. *Developer network: definition of a security vulnerability*. 2018. Disponível em: <https://msdn.microsoft.com/en-us/library/cc751383>. Citado na página 44.

MICROSOFT. *Developer network: diretrizes de codificação segura*. 2018. Disponível em: <https://msdn.microsoft.com/pt-br/library/8a3x2b7f>. Citado 2 vezes nas páginas 45 e 47.

MICROSOFT. *Respondendo a incidentes de segurança de TI*. 2018. Disponível em: <https://technet.microsoft.com/pt-br/library/cc700825.aspx>. Citado na página 50.

MILLER, B. *University of Wisconsin–Madison: Fuzz testing of application reliability*. 2008. Disponível em: <http://pages.cs.wisc.edu/~bart/fuzz/>. Citado na página 48.

MIREK, J. et al. A guide to securing red hat enterprise linux. *RED*, v. 1, 2014. Citado na página 44.

NIST. *National vulnerability database*. 2009. Disponível em: <https://www.nist.gov/programs-projects/national-vulnerability-database-nvd>. Citado na página 35.

NORTHCUTT, S. et al. Penetration testing: Assessing your overall security before attackers do. *SANS Institute Information Security Reading Room*, v. 3, n. 6, p. 4–7, 2006. Citado na página 47.

OTHMANE, L. ben et al. Extending the agile development process to develop acceptably secure software. *IEEE Transactions on Dependable and Secure Computing*, IEEE, v. 11, n. 6, p. 497–509, 2014. Citado 2 vezes nas páginas 61 e 63.

OUNI, A.; KULA, R. G.; INOUE, K. Search-based peer reviewers recommendation in modern code review. In: IEEE. *Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on*. [S.l.], 2016. p. 367–377. Citado na página 47.

OWASP. *CLASP Concepts*. 2010. Disponível em: <https://www.owasp.org/>. Citado 2 vezes nas páginas 53 e 54.

OWASP. *OWASP Top Ten*. 2010. Disponível em: https://www.owasp.org/Category:OWASP_Top_Ten_Project. Citado 6 vezes nas páginas 18, 19, 35, 36, 40 e 51.

OWASP. *Application threat modeling*. 2015. Disponível em: https://www.owasp.org/index.php/Application_Threat_Modeling. Citado 5 vezes nas páginas 39, 41, 42, 43 e 57.

OWASP, F. T. *OWASP Secure coding practices – quick reference guide*. 2012. Disponível em: https://www.owasp.org/images/b/b3/OWASP_SCP_v1.3-pt-BR.pdf. Citado na página 45.

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. Availability, reliability, and security in information systems. *sintef. Availability, Reliability, and Security in Information Systems.*, Conference Publishing Services (CPS), p. 548–555, 2016. Citado 6 vezes nas páginas 60, 61, 63, 65, 66 e 73.

OYETOYAN, T. D.; JAATUN, M. G.; CRUZES, D. S. A lightweight measurement of software security skills, usage and training needs in agile teams. *International Journal of Secure Software Engineering (IJSSE)*, IGI Global, v. 8, n. 1, p. 1–27, 2017. Citado 3 vezes nas páginas 63, 65 e 66.

PALMER, S.; FELSING, J. *A practical guide to Feature-Driven Development*. 1ª. ed. [S.l.]: Prentice Hall, USA, 2002. 35–54 p. Citado 2 vezes nas páginas 27 e 80.

PCI. *Penetration test guidance*. 2015. Disponível em: <https://www.pcisecuritystandards.org>. Citado na página 48.

PETUKHOV, A.; KOZLOV, D. Detecting security vulnerabilities in web applications using dynamic analysis with penetration testing. *Computing Systems Lab, Department of Computer Science, Moscow State University*, OWASP - Application Security Conference, p. 1–16, 2008. Citado 2 vezes nas páginas 45 e 47.

PINHEIRO, J. M. dos S. Ameaças e ataques aos sistemas de informação. UniFOA, p. 1–11, 2011. Citado na página 35.

POPPENDIECK, M.; POPPENDIECK, T. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. [S.l.]: Addison-Wesley, 2003. Citado 2 vezes nas páginas 26 e 80.

POTTER, B.; MCGRAW, G. Building security in: software security testing. *IEEE Security & Privacy*, IEEE, v. 2, n. 5, p. 81–85, 2004. Citado 2 vezes nas páginas 52 e 125.

PRESSMAN, R. *Engenharia de Software*. 7ª. ed. [S.l.]: McGraw Hill, Brasil, 2009. 81–106 p. Citado 6 vezes nas páginas 18, 23, 24, 30, 32 e 33.

RICHARDSON, T.; THIES, C. N. *Secure software design*. [S.l.]: Jones & Bartlett Publishers, 2012. Citado na página 46.

- ROBSON, C.; MCCARTAN, K. *Real world research*. [S.l.]: John Wiley & Sons, 2016. Citado na página 67.
- ROUTH, J. et al. *Appropriate software security control types for third party service and product providers*. 2015. Disponível em: <https://www.fsisac.com/article/appropriate-software-security-control-types-third-party-service-and-product-providers>. Citado na página 55.
- SALANT, P.; DILLMAN, I.; DON, A. *How to conduct your own survey*. [S.l.: s.n.], 1994. Citado na página 67.
- SANTOS, L. R. *Técnicas de teste de software*. 2011. Disponível em: www.inf.ufpr.br/Impres/ci221/aula_tecnicas_teste_Luis_Renato.pdf. Citado na página 47.
- SCHWABER, K.; BEEDLE, M. *Agile software development with Scrum*. [S.l.]: Prentice Hall Upper Saddle River, 2002. v. 1. Citado na página 26.
- SCHWABER, K. S. Um guia definitivo para o SCRUM – As regras do jogo. *Scrum.Org*, Scrum Guide, v. 26, p. 3–15, 2013. Citado 3 vezes nas páginas 30, 31 e 80.
- SEACORD, R. C.; HOUSEHOLDER, A. D. A structured approach to classifying sec. vuln. *CMU SEI*, 2005. Citado na página 35.
- SEI. *SEI- Carnegie mellon university: capability maturity model integration - CMMI*. 2017. Disponível em: <https://www.sei.cmu.edu/cmmi/>. Citado 2 vezes nas páginas 82 e 158.
- SEI, S. E. I. *Team software process*. 2016. Disponível em: <http://www.sei.cmu.edu/tsp/>. Citado na página 54.
- SINDRE, G.; OPDAHL, A. L. Capturing security requirements through misuse cases. *Norsk Informatic Conference*, 2001. Citado na página 39.
- SKARIN, M. *Real-world Kanban: do less, accomplish more with lean thinking*. [S.l.]: The Pragmatic Bookshelf, 2015. Citado 2 vezes nas páginas 27 e 80.
- SODIYA, A. S.; ONASHOGA, S. A.; AJAYI, O. B. Towards building secure software systems. *Issues in Informing Science and Information Technology*, Informing Science Institute, v. 3, p. 635–646, 2006. Citado na página 53.
- SOFTEX. *Melhoria do processo de software brasileiro MPS.br*. 2003. Disponível em: <https://www.softex.br/mpsbr/>. Citado 2 vezes nas páginas 82 e 158.
- SOMMERVILLE, I. *Software EGINEERING*. 9ª. ed. [S.l.]: Pearson Education, Boston, Massachusetts, 2011. 56–81 p. Citado 6 vezes nas páginas 18, 23, 24, 26, 32 e 34.
- SONARSOURCE. *SonarQube: the leading product for continuous code quality*. 2017. Disponível em: <https://www.sonarqube.org/>. Citado na página 120.
- STALLINGS, W. *Computer security*. 2ª. ed. [S.l.]: Pearson Education, 2012. 10–23 p. Citado na página 35.
- STAPLETON, J. Dsdm: Dynamic systems development method. In: IEEE. *Technology of Object-Oriented Languages and Systems, 1999. Proceedings of*. [S.l.], 1999. p. 406–406. Citado na página 27.

SU, Z.; WASSERMANN, G. The essence of command injection attacks in web applications. university of california. In: ACM. *ACM SIGPLAN Notices*. [S.l.], 2006. v. 41, n. 1, p. 372–382. Citado na página 45.

SYMANTEC. *Security response: glossário - buffer overflow: estouro de buffer*. 2016. Disponível em: https://www.symantec.com/pt/br/security_response/glossary/define.jsp?letter=b&word=buffer-overflow. Citado na página 38.

SYMANTEC. *Symantec process: symantec Software Security Process*. 2016. Disponível em: <https://www.symantec.com/content/dam/symantec/docs/about/symantec-software-security-process-en.pdf>. Citado na página 47.

TELLES, R. A efetividade da “matriz de amarração” de mazzon nas pesquisas em administração. *Revista de Administração*, v. 36, n. 4, p. 64–72, 2001. Citado na página 68.

VERACODE. *CA Technologies - Code review: code reviews tools*. 2017. Disponível em: <https://www.veracode.com/security/code-review>. Citado na página 47.

VERDON, D.; MCGRAW, G. Risk analysis. *Security & Privacy*, IEEE, p. 79–84, 2004. Citado na página 35.

VERGARA, S. C. *Métodos de pesquisa em administração*. [S.l.]: Atlas, 2005. Citado na página 75.

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*, Universidade Estadual Paulista-UNESP, Programa de Pós-Graduação em Ciência da Informação, v. 9, n. 1, p. 1–28, 2015. Citado 2 vezes nas páginas 19 e 35.

WÄYRYNEN, J.; BODÉN, M.; BOSTRÖM, G. Security engineering and extreme programming: An impossible marriage? In: SPRINGER. *Conference on Extreme Programming and Agile Methods*. [S.l.], 2004. p. 117–128. Citado 2 vezes nas páginas 61 e 65.

WEBER, T. S. tolerância a falhas: conceitos e exemplos. *Apostila do Programa de Pós-Graduação-Instituto de Informática-UFRGS. Porto Alegre*, p. 1–5, 2003. Citado 2 vezes nas páginas 35 e 36.

WEI, C. Sia: Misuse cases and abuse cases in eliciting security requirements. *System Security: COMPSCI*, v. 725, p. 1–12, 2005. Citado 2 vezes nas páginas 40 e 41.

WEST-BROWN, M. J. et al. *Handbook for computer security incident response teams (csirts)*. [S.l.], 2003. Citado na página 49.

WIESMANN, A. et al. A guide to building secure web applications and web services. *The Open Web Application Security Project*, v. 2, n. 1, p. 31–57, 2005. Citado na página 47.

ZHIOUA, Z.; SHORT, S.; ROUDIER, Y. Static code analysis for software security verification: problems and approaches. In: IEEE. *Computer Software and Applications Conference Workshops COMPSACW 38th International*. [S.l.], 2014. p. 102–109. Citado na página 46.

Apêndice A – Instrumento do *survey*

A.1 Instrumento aplicado aos gerentes de equipes de desenvolvimento ágil de software

Tabela 29 – Grau de implantação e benefícios obtidos com a implantação das técnicas e ferramentas

Nº	Técnicas e Ferramentas	Perguntas	% IMP Pergunta	%IMP Tec. e Ferramentas.	Benefícios Previstos	Benefícios Obtidos
1	Especialista em segurança	1) Existe algum especialista em segurança de software? 2) Foram definidas políticas de segurança para o desenvolvimento ágil de software? 3) Foram definidas responsabilidades de aplicação de técnicas e ferramentas de segurança? 4) As políticas de segurança são aplicadas no projeto?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Especificar políticas de segurança de software para o projeto. 2) Aplicar e acompanhar critérios de políticas de segurança durante o projeto. 3) Melhorar a aplicação e controle de segurança no desenvolvimento de ágil de software. 4) Controlar a aplicação das políticas de segurança e mantê-la nos níveis adequados de segurança.	S/N S/N S/N S/N
1	Requisitos de segurança	1) São especificados requisitos de segurança para a aplicação? 2) Existe algum procedimento para gerenciar esses requisitos? 3) Foi realizada alguma priorização de requisitos de segurança? 4) É utilizada alguma técnica para a coleta de requisitos de segurança?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Identificar os objetivos e restrições de segurança na aplicação. 2) Facilitar a organização e classificação dos requisitos de segurança a fim de manipulá-los quando necessário. 3) Identificar os requisitos mais críticos de segurança. 4) Facilitar a coleta e entendimento dos requisitos de segurança.	S/N S/N S/N S/N

3	<i>Abuse/misuse cases/stories</i>	1) São especificados requisitos de segurança utilizando <i>abuse/misuse cases</i> ? 2) Utiliza algum procedimento para a especificação de requisitos com <i>abuse/misuse cases</i> ? 3) Foi realizada alguma priorização de <i>abuse/misuse cases</i> ? 4) São utilizadas ferramentas para a criação dos <i>misuse/abuse cases</i> ?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Descrever os comportamentos prejudiciais e indesejáveis do sistema. 2) Facilitar a identificação dos possíveis comportamentos prejudiciais e indesejáveis do sistema. 3) Identificar os requisitos mais críticos de segurança. 4) Facilitar o entendimento e difundir as informações obtidas por meio dos <i>abuse/misuse cases/stories</i> à equipe.	S/N S/N S/N S/N
4	Modelagem de ameaça/risco	1) Os requisitos mais críticos são modelados, em relação às possíveis ameaças? 2) São identificadas a probabilidade e impacto das ameaças à segurança? 3) Utiliza alguma estrutura para a modelagem de ameaças? 4) Alguma documentação para modelagem de ameaças é utilizada?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Identificar as ameaças mais críticas e suas contramedidas. 2) Identificar as ameaças de segurança mais críticas, probabilidades e seus efeitos. 3) Identificar e facilitar o entendimento em relação as possíveis ameaças na aplicação. 4) Reportar à equipe a cerca das possíveis ameaças na aplicação e suas contramedidas.	S/N S/N S/N S/N
5	Análise de riscos	1) Existe alguma priorização de requisitos na análise de riscos? 2) São identificadas a probabilidade e impacto dos ataques à segurança? 3) Utiliza alguma estrutura para a análise de riscos? 4) Alguma documentação para a análise de riscos é utilizada?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Facilitar a identificação dos riscos nas funcionalidades mais críticas e suas mitigações. 2) Facilitar a identificação dos riscos de segurança mais críticos, probabilidades e seus efeitos. 3) Identificar e facilitar o entendimento em relação aos possíveis riscos de falta de segurança na aplicação. 4) Reportar à equipe a cerca dos possíveis riscos na aplicação e suas mitigações.	S/N S/N S/N S/N

6	Gestão de respostas a incidentes	<p>1) Existe o gerenciamento de incidentes de segurança?</p> <p>2) São identificados e listados os possíveis incidentes?</p> <p>3) Para cada incidente identificado, existem respostas?</p> <p>4) É utilizada alguma documentação específica para essa gestão?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Facilitar a identificação das ameaças mais críticas e suas contramedidas.</p> <p>2) Facilitar a identificação das ameaças de segurança mais críticas, probabilidades e seus efeitos.</p> <p>3) Identificar e facilitar o entendimento em relação as possíveis ameaças na aplicação.</p> <p>4) Reportar à equipe a cerca das possíveis ameaças na aplicação e suas contramedidas.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
7	Design de segurança	<p>1) Realiza a identificação de componentes críticos da aplicação?</p> <p>2) São aplicadas políticas de segurança no <i>Design</i> da aplicação?</p> <p>3) Utiliza a modelagem de ameaças e riscos para se basear no <i>Design</i>?</p> <p>4) É utilizada alguma notação específica para <i>Design</i> seguro?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Facilitar a identificação dos componentes críticos da aplicação.</p> <p>2) Aplicar critérios de segurança nos componentes da aplicação.</p> <p>3) Facilitar a aplicação de critérios de segurança por meio da utilização da modelagem de ameaças e riscos.</p> <p>4) Facilitar o entendimento do <i>Design</i> em relação a segurança.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
8	Contramedidas de segurança	<p>1) Existe o gerenciamento dos incidentes de segurança?</p> <p>2) São identificados e listados os possíveis incidentes de segurança?</p> <p>3) Para cada incidente identificado, são identificadas as contramedidas?</p> <p>4) É utilizada alguma documentação específica para o gerenciamento de contramedidas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Organizar as contramedidas de segurança para cada ameaça identificada.</p> <p>2) Introduzir proteção contra possíveis ameaças.</p> <p>3) Facilitar a identificação de contramedidas de segurança para cada ameaça identificada.</p> <p>4) Difundir a informação relacionada às ameaças e suas contramedidas para a equipe.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>

9	Codificação segura	1) Utiliza práticas de codificação segura? 2) Utiliza técnicas de revisão de código? 3) Aplica políticas de segurança durante a codificação? 4) Utiliza ferramentas de segurança durante a codificação?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Difundir a segurança na codificação, por meio da utilização de compiladores recentes e organizando listas com funções inseguras, por exemplo. 2) Analisar se as práticas de codificação segura vêm sendo aplicadas. 3) Assegurar se as práticas de segurança que foram definidas inicialmente vêm sendo aplicadas. 4) Facilitar a aplicação de práticas de codificação segura.	S/N S/N S/N S/N
10	Ferramenta de modelagem de ameaças	1) São utilizadas ferramentas para modelar as possíveis ameaças? 2) São utilizadas ferramentas para calcular a probabilidade e impacto das ameaças? 3) Utiliza alguma ferramenta para gerenciar as ameaças? 4) As ferramentas utilizadas identificam as contramedidas e mitigações?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Facilitar a identificação das ameaças mais críticas e suas contramedidas. 2) Facilitar a identificação das ameaças de segurança mais críticas, probabilidades e seus efeitos. 3) Identificar e facilitar o entendimento em relação as possíveis ameaças na aplicação. 4) Reportar à equipe a cerca das possíveis ameaças na aplicação e suas contramedidas.	S/N S/N S/N S/N
11	Análise estática de código	1) São utilizadas ferramentas para analisar defeitos estaticamente no código-fonte? 2) Tais ferramentas reportam os possíveis defeitos? 3) Identificam quais trechos de código são mais suscetíveis a defeitos? 4) As ferramentas utilizadas criam uma lista dos defeitos identificados?	0 até 1 0 até 1 0 até 1 0 até 1	0% até 100%	1) Detectar e facilitar a identificação de defeitos diretamente no código-fonte. 2) Reportar à equipe informações a cerca dos possíveis defeitos no código-fonte e suas soluções. 3) Identificar e facilitar o entendimento em relação aos trechos do código-fonte mais suscetíveis a apresentarem defeito. 4) Facilitar a identificação dos defeitos mais críticos, probabilidades de ocorrência e seus efeitos a fim de manter um guia com boas práticas.	S/N S/N S/N S/N

12	Análise dinâmica de código	<p>1) São utilizadas ferramentas para analisar dinamicamente a aplicação?</p> <p>2) Tais ferramentas reportam as possíveis vulnerabilidades?</p> <p>3) Identificam quais funcionalidades são mais suscetíveis a vulnerabilidades?</p> <p>4) As ferramentas utilizadas criam uma lista das vulnerabilidades identificadas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Facilitar a identificação de defeitos no comportamento da aplicação.</p> <p>2) Reportar à equipe informações a cerca dos possíveis defeitos no comportamento da aplicação e suas soluções.</p> <p>3) Identificar e facilitar o entendimento em relação às funcionalidades mais suscetíveis a apresentarem defeito.</p> <p>4) Facilitar a identificação dos defeitos mais críticos, probabilidades de ocorrência e seus efeitos.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
13	Ferramenta de revisão de código	<p>1) São utilizadas ferramentas para analisar estaticamente o código-fonte?</p> <p>2) Tais ferramentas reportam as possíveis vulnerabilidades?</p> <p>3) Identificam quais trechos de código são mais suscetíveis a vulnerabilidades?</p> <p>4) As ferramentas utilizadas criam uma lista das vulnerabilidades identificadas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Facilitar a identificação de defeitos diretamente no código-fonte.</p> <p>2) Reportar à equipe informações a cerca dos possíveis defeitos no código-fonte e suas soluções.</p> <p>3) Identificar e facilitar o entendimento em relação aos trechos do código-fonte mais suscetíveis a apresentarem defeito.</p> <p>4) Facilitar a identificação dos defeitos mais críticos, probabilidades de ocorrência e seus efeitos.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
14	Avaliação de vulnerabilidades	<p>1) As vulnerabilidades identificadas são gerenciadas?</p> <p>2) Tais vulnerabilidades são classificadas e organizadas por prioridades?</p> <p>3) As listas de vulnerabilidades são atualizadas e reclassificadas?</p> <p>4) Para cada vulnerabilidade são identificadas as ferramentas para detecção destas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Fornecer informações de segurança, atualizar e educar a equipe em relação as consequências das vulnerabilidades mais comuns encontradas em aplicações web.</p> <p>2) Atualizar lista de vulnerabilidades atualizadas, suas probabilidades de ocorrência e seus efeitos.</p> <p>3) Manter lista de contramedidas para as vulnerabilidades identificadas e manter um guia atualizado para a equipe.</p> <p>4) Utilizar ferramentas adequadas para cada tipo de vulnerabilidades identificadas a fim de ter efetividade no procedimento.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>

15	Teste de penetração	<p>1) São utilizadas ferramentas para teste de penetração?</p> <p>2) Tais ferramentas identificam as possíveis vulnerabilidades?</p> <p>3) Tais ferramentas realizam as tentativas de acesso a aplicação?</p> <p>4) As ferramentas utilizadas detalham as vulnerabilidades identificadas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Determinar como um usuário mal-intencionado pode obter acesso não autorizado aos ativos da organização.</p> <p>2) Identifica as vulnerabilidades para determinar se é possível realizar o acesso não autorizado no sistema.</p> <p>3) Explorar as vulnerabilidades para determinar se é possível realizar o acesso não autorizado no sistema.</p> <p>4) Manter uma lista com as vulnerabilidades identificadas no teste de penetração.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
16	<i>Fuzz testing</i>	<p>1) São utilizadas ferramentas para <i>Fuzz Testing</i>?</p> <p>2) Tais ferramentas geram as entradas aleatórias e inválidas necessárias?</p> <p>3) Tais ferramentas realmente exploram as vulnerabilidades?</p> <p>4) As ferramentas utilizadas detalham as vulnerabilidades identificadas?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Observar o comportamento após geração de entradas aleatórias ou inesperadas.</p> <p>2) Observar os possíveis defeitos, como lançamento de exceções imprevistas, colapso do cliente ou servidor, entre outros.</p> <p>3) Analisar os efeitos de uma falha ao inserir entradas inválidas ou aleatórias, o que constatará que o teste foi bem sucedido.</p> <p>4) Utilizar informações obtidas para desenvolver contramedidas de segurança.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>
17	Teste baseado em risco	<p>1) São realizados testes baseados em risco?</p> <p>2) Os testes foram baseados na documentação da modelagem de riscos?</p> <p>3) Os testes exploram cada risco?</p> <p>4) Os riscos são acompanhados e controlados?</p>	<p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p> <p>0 até 1</p>	<p>0% até 100%</p>	<p>1) Identificar os aspectos que podem prejudicar o funcionamento correto da aplicação.</p> <p>2) Facilitar e agilizar os testes baseados em riscos.</p> <p>3) Executar os testes a partir dos requisitos de segurança e modelagem de riscos.</p> <p>4) gerenciar a probabilidade e impacto de cada risco, bem como testá-los e priorizá-los.</p>	<p>S/N</p> <p>S/N</p> <p>S/N</p> <p>S/N</p>

18	Revisão de segurança de código	1) São realizadas revisões de segurança de código?			1) Detectar e facilitar a identificação de defeitos relacionados a segurança diretamente no código-fonte.	
		2) Tais revisões reportam as possíveis vulnerabilidades?	0 até 1		2) Reportar à equipe informações os possíveis defeitos relacionados a segurança no código-fonte e suas soluções.	S/N
		3) Identificam quais trechos de código são mais suscetíveis a vulnerabilidades?	0 até 1	0% até	3) Identificar e facilitar o entendimento em relação aos trechos do código-fonte mais suscetíveis a defeito de segurança.	S/N
		4) A revisão de segurança lista as vulnerabilidades identificadas e suas mitigações?	0 até 1	100%	4) Facilitar a identificação dos defeitos de segurança mais críticos, ocorrência e seus efeitos.	S/N

Fonte: Lígia Santos, 2017

A.2 Instrumento aplicado aos membros de equipes de desenvolvimento ágil de software

Um estudo de caso sobre o uso de técnicas e ferramentas para detecção de vulnerabilidades em organizações que utilizam métodos ágeis.

Introdução.

Este questionário tem por objetivo identificar, descrever e comparar o estado da prática na utilização de técnicas e ferramentas de detecção de vulnerabilidades em desenvolvimento ágil de software aplicadas em organizações brasileiras desenvolvedoras de software que utilizam métodos ágeis.

O questionário está dividido em quatro seções. A Seção A aborda informações gerais sobre os projetos. A Seção B relaciona a aptidão, interesses e uso atual de técnicas e ferramentas de detecção de vulnerabilidades e, finalmente, a Seção C está focada em perguntas relacionadas ao treinamento e necessidade de treinamento e, finalmente, a Seção D aborda informações gerais sobre a implantação das técnicas e ferramentas.

Obrigado por participar dessa pesquisa. Ao iniciá-la, você estará aceitando todos os termos e condições.

Procedimentos para participação.

Para concluir este questionário, é necessário seguir os seguintes procedimentos:

- ✓ O Sr. (a) deve ser membro da equipe de desenvolvimento de software.
- ✓ O questionário será aplicado por meio de um formulário online ou físico que será disponibilizado aos participantes, sem qualquer necessidade de deslocamento das pessoas envolvidas.
- ✓ Cada um dos participantes receberá apenas um formulário composto por 18 questões.
- ✓ Todas as questões são de preenchimento obrigatório.
- ✓ No experimento serão apresentadas técnicas e ferramentas para detecção de vulnerabilidades no desenvolvimento ágil de software.
- ✓ O participante deverá escolher apenas uma alternativa.
- ✓ A aplicação do questionário tem duração de 20 minutos.

Seção A: Informações gerais sobre o projeto mais relevante.

As perguntas a seguir têm por objetivo levantar informações sobre a implantação de técnicas e ferramentas para detecção de vulnerabilidades nas organizações em projetos.

1) Qual sua função na equipe de desenvolvimento ágil? *

- ☐ Especialista em segurança
- ☐ Analista de requisitos
- ☐ Arquiteto de software
- ☐ Desenvolvedor
- ☐ Testador
- ☐ Outro (Especificar)

2) Qual (is) método (s) ágil (is) você utiliza? *

- ☐ Scrum
- ☐ Extreme Programming (XP)
- ☐ Feature Driven Development (FDD)
- ☐ Lean Software Development
- ☐ Crystal Methods
- ☐ Kanban
- ☐ Agile Unified Process (AUP)
- ☐ Dynamics Systems Development Method (DSDM)
- ☐ Outro (Especificar)

3) Quantos anos trabalha na atual função na equipe de desenvolvimento de software? *

- ☐ 0 a 1 ano.
- ☐ 2 anos a 5 anos.
- ☐ 6 anos a 10 anos.
- ☐ Acima de 10 anos.

4) Com qual tipo de tecnologia atua? *

- ☐ Web
- ☐ Mobile
- ☐ Rede
- ☐ E-commerce
- ☐ Outro (Especificar)

5) Qual o tamanho da equipe que implantou as técnicas e ferramentas para detecção de vulnerabilidades no projeto mais relevante? *

6) Adotou-se alguma metodologia de implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto? *

- ☐ CMMI
- ☐ MPS.Br
- ☐ ITIL
- ☐ Metodologia própria
- ☐ Nenhuma
- ☐ Outra (Especificar)

7) Responsável pelo desenvolvimento de atividades de segurança no projeto mais relevante: *

- ☐ Especialista em segurança
- ☐ Analista de requisitos
- ☐ Arquiteto de software
- ☐ Desenvolvedor
- ☐ Testador
- ☐ Não existe atribuição formal para essa responsabilidade
- ☐ Outros (Especificar)

Escala	Descrição dos níveis de habilidades
Baixo	Não possui experiência de trabalho nesta área.
Básico	Possui nível de experiência adquirida em cursos. Necessita de ajuda ao executar as tarefas.
Moderado	É capaz de completar com sucesso as tarefas nessa área. Pode solicitar o auxílio de um especialista, mas consegue executar as habilidades de forma independente.
Alto	É capaz de executar as tarefas associadas à área sem auxílio de terceiros. Pode ser reconhecido dentro da organização como um consultor imediato.
Especialista	É conhecido como um especialista. Possui capacidade de fornecer orientação, solucionar problemas e responder a perguntas relacionadas com esta área.

8) Qual seu grau de aptidão nessas técnicas ou ferramentas? *

	Baixa	Básica	Moderada	Alta	Especialista
Especialista em segurança	☐	☐	☐	☐	☐
Requisitos de segurança	☐	☐	☐	☐	☐
Abuse/Misuse Case/Stories	☐	☐	☐	☐	☐
Modelagem de ameaças	☐	☐	☐	☐	☐
Modelagem de riscos	☐	☐	☐	☐	☐
Gestão de respostas a incidentes	☐	☐	☐	☐	☐
Design de segurança	☐	☐	☐	☐	☐
Contramedidas de segurança	☐	☐	☐	☐	☐
Codificação segura	☐	☐	☐	☐	☐
Ferramenta de modelagem de ameaças/riscos	☐	☐	☐	☐	☐
Ferramenta de análise estática de código	☐	☐	☐	☐	☐
Ferramenta de análise dinâmica de código	☐	☐	☐	☐	☐
Ferramenta de revisão de código	☐	☐	☐	☐	☐
Avaliação de vulnerabilidades	☐	☐	☐	☐	☐
Teste de penetração	☐	☐	☐	☐	☐
Fuzz testing	☐	☐	☐	☐	☐
Teste baseado em riscos	☐	☐	☐	☐	☐
Revisão de segurança de código	☐	☐	☐	☐	☐

9) Qual seu grau de interesse nessas técnicas ou ferramentas? *

	Não interessado	Levemente interessado	Moderadamente interessado	Interessado	Muito interessado
Especialista em segurança	☐	☐	☐	☐	☐
Requisitos de segurança	☐	☐	☐	☐	☐
Abuse/Misuse Case/Stories	☐	☐	☐	☐	☐
Modelagem de ameaças	☐	☐	☐	☐	☐
Modelagem de riscos	☐	☐	☐	☐	☐
Gestão de respostas a incidentes	☐	☐	☐	☐	☐
Design de segurança	☐	☐	☐	☐	☐
Contramedidas de segurança	☐	☐	☐	☐	☐
Codificação segura	☐	☐	☐	☐	☐
Ferramenta de modelagem de ameaças/riscos	☐	☐	☐	☐	☐
Ferramenta de análise estática de código	☐	☐	☐	☐	☐
Ferramenta de análise dinâmica de código	☐	☐	☐	☐	☐
Ferramenta de revisão de código	☐	☐	☐	☐	☐
Avaliação de vulnerabilidades	☐	☐	☐	☐	☐
Teste de penetração	☐	☐	☐	☐	☐
Fuzz testing	☐	☐	☐	☐	☐
Teste baseado em riscos	☐	☐	☐	☐	☐
Revisão de segurança de código	☐	☐	☐	☐	☐

10) Você utiliza alguma dessas técnicas ou ferramentas? *

	Sim	Não
Especialista em segurança	☐	☐
Requisitos de segurança	☐	☐
Abuse/Misuse Case/Stories	☐	☐
Modelagem de ameaças	☐	☐
Modelagem de riscos	☐	☐
Gestão de respostas a incidentes	☐	☐
Design de segurança	☐	☐
Contramedidas de segurança	☐	☐
Codificação segura	☐	☐
Ferramenta de modelagem de ameaças/riscos	☐	☐
Ferramenta de análise estática de código	☐	☐
Ferramenta de análise dinâmica de código	☐	☐
Ferramenta de revisão de código	☐	☐
Avaliação de vulnerabilidades	☐	☐
Teste de penetração	☐	☐
Fuzz testing	☐	☐
Teste baseado em riscos	☐	☐
Revisão de segurança de código	☐	☐

Seção C: Treinamento e necessidade de treinamento.

As perguntas a seguir têm por objetivo levantar informações em relação às necessidades de treinamento e interesse em receber treinamento nas técnicas e ferramentas para detecção de vulnerabilidades nas organizações estudadas.

11) Responda sim ou não para o conjunto de técnicas ou ferramentas que tem interesse em receber treinamento. *

	Sim	Não
Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	☐	☐
Design de segurança, contramedidas de segurança e codificação segura.	☐	☐
Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	☐	☐
Avaliação de vulnerabilidades, teste de penetração e Fuzz testing.	☐	☐
Teste baseado em riscos e revisão de segurança de código.	☐	☐

12) Responda sim ou não para o conjunto de técnicas ou ferramentas que você tem necessidade em receber treinamento. *

	Não Interessado	Levemente Interessado	Moderadamente Interessado	Muito Interessado	Não sei
Especialista em segurança, requisitos de segurança e Abuse/Misuse Cases/Stories.	☐	☐	☐	☐	☐
Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.	☐	☐	☐	☐	☐
Design de segurança, contramedidas de segurança e codificação segura.	☐	☐	☐	☐	☐
Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.	☐	☐	☐	☐	☐
Avaliação de vulnerabilidades, teste de penetração e Fuzz testing.	☐	☐	☐	☐	☐
Teste baseado em riscos e revisão de segurança de código.	☐	☐	☐	☐	☐

Seção D: Perguntas gerais sobre a implantação das técnicas e ferramentas.

As perguntas a seguir serão têm por objetivo levantar informações em relação à implantação das técnicas e ferramentas para detecção de vulnerabilidades nas atividades de desenvolvimento ágil de software realizadas nas organizações estudadas.

13) Qual a motivação para a implantação de segurança no desenvolvimento ágil de software? *

14) Por que foram escolhidas essas técnicas e ferramenta de detecção de vulnerabilidades? *

15) Como foi realizada a implantação dessas técnicas e ferramentas? *

16) Qual (is) estratégia (s) utilizada (s) para a implantação dessas técnicas e ferramentas de detecção de vulnerabilidades? *

17) Qual (is) a (s) dificuldade (s) e limitação (es) encontrada na implantação de atividades de segurança? *

18) Qual (is) a (s) lição (es) aprendida (s) com a implantação de atividades de segurança compostas por técnicas e ferramentas para detecção de vulnerabilidades? *

A.3 Roteiro semi-estruturado utilizado para condução das entrevistas

Condução da entrevista com os gerentes

Introdução:

Apresentar a equipe que conduzirá a entrevista.

Agradecer a participação.

Assegurar a confidencialidade da entrevista.

Pedir permissão para gravar a entrevista.

Condução:

***Sanar possíveis dúvidas referentes ao questionário de técnicas e ferramentas de detecção de vulnerabilidades.**

1- Qual sua função na equipe de desenvolvimento ágil?

2- Qual (is) método (s) ágil (eis) você utiliza?

3- Trabalha há quantos anos na atual função na equipe de desenvolvimento de software?

4- Com qual tipo de tecnologia atua?

5- Qual o tamanho da equipe que implantou as técnicas e ferramentas para detecção de vulnerabilidades no projeto?

6- Adotou-se alguma metodologia de implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto?

7- Quem é o responsável pelo desenvolvimento de atividades de segurança no projeto mais relevante?

8- Qual seu grau de aptidão, interesses na técnica e ferramentas e uso atual?

-Considere para aptidão a escala: baixa, básica, moderada, alta e especialista.

-Considere para interesse a escala: não interessado, levemente interessado, moderadamente interessado, interessado e muito interessado.

-Considere para uso atual: sim ou não.

- TF01- Especialista em segurança
- TF02- Requisitos de segurança
- TF03- *Abuse/Misuse Case/Stories*
- TF04- Modelagem de ameaças
- TF05- Modelagem de riscos
- TF06- Gestão de respostas a incidentes
- TF07- Design de segurança
- TF08- Contramedidas de segurança
- TF09- Codificação segura
- TF10- Ferramenta de modelagem de ameaças/riscos

- TF11- Ferramenta de análise estática de código
- TF12- Ferramenta de análise dinâmica de código
- TF13- Ferramenta de revisão de código
- TF14- Avaliação de vulnerabilidades
- TF15- Teste de penetração
- TF16- *Fuzz testing*
- TF17- Teste baseado em riscos
- TF18- Revisão de segurança de código

9- Qual o conjunto de técnicas ou ferramentas que tem interesse e necessidade em receber treinamento? (As Técnicas e ferramentas foram agrupadas conforme a ordem de utilização no ciclo de vida de desenvolvimento de software)

- TF01- Especialista em segurança, requisitos de segurança e *abuse/misuse case/stories*.
- TF02- Modelagem de ameaças, modelagem de riscos e gestão de respostas a incidentes.
- TF03- *Design* de segurança, contramedidas de segurança e codificação segura.
- TF04- Ferramenta de modelagem de ameaças/riscos, ferramenta de análise estática de código, ferramenta de análise dinâmica de código e ferramenta de revisão de código.
- TF05- Avaliação de vulnerabilidades, teste de penetração e *Fuzz testing*.
- TF06- Teste baseado em riscos e revisão de segurança de código

10- Qual a motivação para a implantação de segurança no desenvolvimento ágil de software?

11- Por que foram escolhidas essas técnicas e ferramenta de detecção de vulnerabilidades?

12- Como foi realizada a implantação dessas técnicas e ferramentas?

13- Qual (is) estratégia (s) utilizada (s) para a implantação dessas técnicas e ferramentas de detecção de vulnerabilidades?

14- Qual (is) a (s) dificuldade (s) e limitação (es) encontrada na implantação de atividades de segurança?

15- Qual (is) a (s) lição (es) aprendida (s) com a implantação de atividades de segurança compostas por técnicas e ferramentas para detecção de vulnerabilidades?

Encerramento:

Alguma pergunta?

Existe alguma outra coisa que gostaria de discutir que não foi tratada pelas perguntas?

Feedback sobre a entrevista.

A.4 Exemplos de técnicas e ferramentas para detecção de vulnerabilidades

Quadro 7 – Exemplos de técnicas e ferramentas

Técnicas e Ferramentas	Exemplos
Requisitos de segurança	<i>UMLSec, Abuse cases/stories, Misuse cases/stories, etc.</i>
Modelagem de ameaças/risco	Árvore de ameaças, STRIDE, DREAD, etc.
Técnicas de contramedidas de segurança	Políticas de segurança, lista de vulnerabilidades e ameaças, BSIMM, etc.
Análise de riscos	OCTAVE, STRIDE, DREAD, etc.
<i>Abuse/misuse case/stories</i>	<i>UMLSec, Secure UML, UML, etc.</i>
Ferramenta de modelagem de ameaça	AS/NZS 4360:2004, STRIDE, DREAD, etc.
Revisão de código	Políticas de segurança, guias de boas práticas, ferramentas específicas, etc.
Gestão de respostas a incidentes	Microsoft Security Response Center - MSRC. Computer Security Incident Handling Guide - NIST. SANS Institute - The Incident Handlers Handbook.
<i>Design</i> de segurança	<i>UMLSec, Secure UML, Árvore de ameaças, etc.</i>
Codificação segura	Revisão de código, bibliotecas seguras, programação em pares, etc.
Análise estática de código	<i>Parsing</i> , Análise de fluxo de dados, Análise de fluxo de controle, etc.
Análise dinâmica de código	Teste de penetração, <i>Fuzz testing</i> , Injeção de falha, etc.
Ferramenta de revisão de código	<i>FindBugs, FindSecBugs</i> , Codacy, etc.
Avaliação de vulnerabilidades	Injeções SQL, Estouro de memória, Força Bruta, <i>Fuzz testing</i> , etc.
Teste de penetração	Netcraf, Httpprint, WebRecon, etc.
<i>Fuzz testing</i>	OWASP WSFuzzer, wFUZER
Teste baseado em risco	<i>Inside-Out, Outside-in</i> , Taxonomy Based questionnaire – TBQ, etc.
Revisão de segurança de código	Kiuwan, <i>Seeker</i> , Codacy, etc.

Fonte: Lígia Santos, 2017

Apêndice B – Mais características da aplicação das técnicas e ferramentas para detecção de vulnerabilidades

As características da implantação das técnicas e ferramentas para detecção de vulnerabilidades foram obtidas por meio de questões abertas de 15 gerentes de equipes de desenvolvimento de software. As perguntas tratavam das motivações, dificuldades, limitações e lições aprendidas na sua implantação. Todas as perguntas tratadas nesta seção passaram pela técnica de análise de conteúdo conforme descrito na Seção 5.

Tabela 30 – Motivação da implantação

Motivações apontadas pelos respondentes	Quantidade %
Garantia de segurança do software desenvolvido	20,00
Qualidade do software desenvolvido	13,33
Recomendação ou exigência do cliente	6,67
Manutenção ou reuso de componentes em outras aplicações	6,67
Aumento dos lucros da organização	6,67
Branco	40,00
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	6,67
Total	100%

Fonte: Lígia Santos, 2017

Ao serem questionados sobre as motivações da implantação de técnicas e ferramentas para detecção de vulnerabilidades, 20% dos gerentes citaram a garantia de segurança do software desenvolvido. A segurança de software é um aspecto que deve ter sua devida atenção, independentemente do tipo de projeto, pois uma aplicação vulnerável pode causar prejuízos as organizações.

A qualidade de software foi apontada por 13,33% dos gerentes como a motivação para a implantação de técnicas e ferramentas para detecção de vulnerabilidades. Para eles, se o software for seguro, confiável e íntegro, ele possui qualidade. Sendo assim, a segurança pode ser considerada um aspecto que influencia na qualidade, porém não diretamente.

A segurança de software deve ter uma atenção especial, analisando cuidadosamente os possíveis pontos que podem influenciar negativamente a funcionalidade da aplicação e suas mitigações.

A recomendação ou exigência do cliente motivou a implantação para 6,67% dos gerentes. Sendo assim, o cliente solicitou que determinadas técnicas e ferramentas fossem

implantadas a fim de proporcionar segurança no software produzido. Essa recomendação ou exigência pode estar relacionada a área de atuação do cliente.

Constantes manutenções causadas pela ausência segurança e impossibilidade de reúso de componentes em outros softwares motivou os envolvidos no projeto a considerarem a implantação de segurança algo relevante. Sendo assim, 6,67% dos gerentes apontaram que constantes manutenções originadas por problemas de segurança motivou os membros de equipes considerarem segurança em seus projetos. Um dos benefícios dessa ação é a possibilidade de reutilização de componentes considerados seguros em outras aplicações.

Aumento do lucro da organização foi uma das motivações tratadas por 6,67% dos gerentes. Se os clientes exigirem que um software confiável seja produzido, a equipe deve atender às especificações. Logo, eles podem ser considerados softwares de qualidade. Um software de qualidade atenderá às necessidades de seus clientes, o que poderá aumentar seus lucros.

40% dos gerentes não especificaram as motivações da implantação das técnicas e ferramentas para detecção de vulnerabilidades. Já 6,67% não implantaram técnicas e ferramentas.

Os porquês da escolha das técnicas e ferramentas dd detecção de vulnerabilidades a serem implantadas podem ser visualizados na Tabela 31

Tabela 31 – Escolhas das técnicas e ferramentas para detecção de vulnerabilidades

Escolha das técnicas e ferramentas	Quantidade %
Facilidade e bom desempenho dos membros da equipe no uso das técnicas e ferramentas	20,00
Cultura da organização	13,33
Qualidade do software desenvolvido	6,67
Branco	46,67
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	13,33
Total	100%

Fonte: Lígia Santos, 2017

As técnicas e ferramentas utilizadas pelos gerentes em seus projetos foram escolhidas de acordo com o bom desempenho e facilidade dos membros em manuseá-las. Sendo assim, 20% dos gerentes especificaram esse porquê na escolha das técnicas e ferramentas a serem implantadas.

A organização pode incentivar o uso de determinadas técnicas e ferramentas em seus projetos, ou por estarem em conformidade com o orçamento da organização, pelo fato de terem melhor desempenho na detecção de vulnerabilidades, ou ainda pelos membros terem um melhor desempenho na sua utilização. Dessa forma, a organização incorpora em sua cultura o uso constante dessas técnicas e ferramentas. Essa escolha foi apontada por 13,33% dos gerentes.

A qualidade foi apontada por 6,67% dos gerentes como um porquê na escolha das técnicas e ferramentas a serem implantadas. As técnicas e ferramentas para detecção de vulnerabilidades são escolhidas com a finalidade de agregar qualidade ao software desenvolvido.

Os gerentes apontaram a qualidade do software produzido como prioridade. Porém, um software com qualidade deve atender às especificações do cliente, a segurança pode ou não ser uma delas. 46,67% dos gerentes não responderam quais estratégias foram utilizadas para a implantação das técnicas e ferramentas para detecção de vulnerabilidades nos projetos. Já para 13,33%, a implantação das técnicas e ferramentas para detecção de vulnerabilidades não foi realizada.

Os detalhes da implantação das técnicas e ferramentas estão apresentadas na Tabela 32

Tabela 32 – Detalhes da implantação das técnicas e ferramentas

Detalhes da implantação	Quantidade %
Treinamento e difusão do conhecimento por parte dos membros da equipe	13,33
De forma incremental	13,33
Por meio de metodologias existentes ou próprias	6,67
Branco	53,33
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	6,67
Não sei	6,67
Total	100%

Fonte: Lígia Santos, 2017

Para 13,33% a implantação foi realizada por meio de treinamento e difusão do conhecimento das técnicas e ferramentas entre membros. Esse treinamento foi baseado nas técnicas e ferramentas a serem implantadas no projeto. Dessa forma, os membros mais experientes organizaram treinamentos aos demais membros. Além de ser um meio

mais barato e cômodo de difundir o conhecimento, ela proporciona maior segurança aos membros de equipes, já que eles são incentivados pelos seus próprios companheiros a usarem as técnicas e ferramentas para detecção de vulnerabilidades nos projetos em que atuam.

A implantação foi realizada de forma incremental para 13,33% dos gerentes. As técnicas e ferramentas foram implantadas a medida que surgiam as necessidades. Dessa forma, apenas algumas técnicas e ferramentas foram implantadas em determinadas fases do ciclo de desenvolvimento ágil de software, gradativamente.

Para 6,67% dos gerentes, a implantação ocorreu por meio do uso de metodologias existentes ou próprias. Uma metodologia pode auxiliar na implantação das técnicas e ferramentas. Exemplos são o ITIL (*Information Technology Infrastructure Library*) (FREITAS, 2013), MPS.br (Melhoria do Processo de Software Brasileiro) (SOFTEX, 2003) e CMMI (*Capability Maturity Model Integration*) (SEI, 2017).

Ainda 53,33% dos gerentes não especificaram como foi realizada a implantação das técnicas e ferramentas. 6,67% dos gerentes alegaram não terem realizado implantação de técnicas e ferramentas.

Finalmente, 6,67% dos gerentes alegaram não saber como foi realizada a implantação. As estratégias da implantação das técnicas e ferramentas estão apresentadas na Tabela 33.

Tabela 33 – Estratégias utilizadas para a implantação das técnicas e ferramentas para detecção de vulnerabilidades

Estratégias de implantação das técnicas e ferramentas	Quantidade %
Definição de políticas de segurança no desenvolvimento de software	6,67
Comparação de casos de sucesso em relação à utilização de segurança	6,67
Utilização de técnicas e ferramentas em projetos isolados	6,67
Treinamento e difusão do conhecimento entre membros da equipe	6,67
Branco	53,33
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	20,00
Total	100%

Fonte: Lígia Santos, 2017

Para 6,67% dos gerentes a estratégia utilizada para a implantação das técnicas e ferramentas ocorreu por meio da definição de políticas de segurança. Dessa forma, os gerentes definiram políticas a serem seguidas no projeto. Elas especificavam quais técnicas

e ferramentas deviam ser implantadas em cada uma das fases do ciclo de desenvolvimento ágil de software.

A busca por ocorrências de implantações de segurança bem-sucedidas foi apontado por 6,67% dos gerentes. Sendo assim, foram analisados projetos relevantes onde foram utilizadas técnicas e ferramentas para detecção de vulnerabilidades e analisadas quais proporcionaram mais benefícios, quais obtiveram melhores desempenho na busca por vulnerabilidades, quais possuem melhores custos e mais fáceis de serem utilizadas. Após essa comparação, a gerência e demais membros selecionaram as técnicas e ferramentas que seriam utilizadas no projeto e realizaram a sua implantação.

Para 6,67% dos gerentes a estratégia ocorreu por meio da implantação das técnicas e ferramentas em projetos isolados. Ou seja, os gerentes optaram por implantar segurança gradativamente nos projetos. Eram realizadas implantações em determinadas fases do ciclo de vida ágil e analisados seus benefícios. Visto que a implantação foi benéfica, ela ocorria nas demais fases até que todo o ciclo de vida tivessem as técnicas e ferramentas implantadas.

O treinamento e difusão do conhecimento entre membros de equipe foi a estratégia utilizada por 6,67% dos gerentes. Visto que algumas equipes eram compostas por membros experientes, os gerentes recomendaram que houvesse o compartilhamento do conhecimento entre demais membros, o *feedback* constante, a colaboração, a interação e o incentivo do uso das técnicas e ferramentas para detecção de vulnerabilidades nos projetos. Dessa forma os membros menos experientes adquiriam maturidade na implantação, já que os mais experientes já haviam realizado tais atividades.

Já 53,33% dos gerentes não responderam quais estratégias utilizaram para implantar as técnicas e ferramentas de detecção e vulnerabilidades nos projetos. Já 20% dos gerentes não realizaram implantação de técnicas e ferramentas.

As dificuldades e limitações ocorridas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 34.

Tabela 34 – Dificuldades e limitações encontradas na implantação de atividades de segurança

Dificuldades e limitações apontadas pelos respondentes	Quantidade %
Dificuldades e limitações relacionados à equipe	20,00
Cultura organizacional não sensibilizada para segurança	6,67
Não há dificuldades	6,67
Aplicação das técnicas e ferramentas nos projetos	6,67
Branco	46,67
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	13,33
Total	100%

Fonte: Lígia Santos, 2017

Como é possível observar, 20% dos gerentes citaram que as dificuldades e limitações foram relacionadas a equipe. Elas podem ser ocasionadas pela falta de conhecimento técnico, falta de interação entre membros especialistas e demais membros, resistência e desinteresse no uso da técnica e ferramenta. Sendo assim, é importante que haja entrosamento entre membros e gerência, incentivo da implantação de segurança nos projetos e treinamento. Utilizar alguma estratégia de implantação, como mostra a Tabela 33, contribui para amenizar a ocorrência dessas dificuldades e limitações na implantação.

A cultura organizacional não sensibilizada para segurança foi um fator que dificultou e limitou a implantação para 6,67% dos gerentes. Dessa forma, a organização pode desconhecer a existência de técnicas e ferramentas para detecção de vulnerabilidades, bem como a necessidade de se ter segurança nos projetos. Pelo fato de a organização não estar habituada com essas técnicas e ferramentas, os membros e gerência podem estar impossibilitados de prosseguir com a implantação.

Para 6,67% dos gerentes não ocorreram dificuldades e limitações durante a implantação das técnicas e ferramentas para detecção de vulnerabilidades nos projetos.

A maior dificuldade e limitação apontada por 6,67% dos gerentes, foi em relação ao manuseio das técnicas e ferramentas. Dificuldade em planejar e implantar as técnicas e ferramentas e utilizá-las em cada uma das fases do ciclo de vida de desenvolvimento ágil de software. Ou seja, quais as técnicas e ferramentas que melhor se adequam em cada uma das fases, como utilizá-las sem alterar a essência dos métodos ágeis e quais são compatíveis com outras tecnologias.

Assim, 46,67% dos gerentes não responderam quais foram as dificuldades e limitações ocorridas durante a implantação. Finalmente, 13,33% dos gerentes apontaram que não há implantação das técnicas e ferramentas para detecção de vulnerabilidades no projeto.

As lições aprendidas durante a implantação das técnicas e ferramentas estão apresentadas na Tabela 35.

Tabela 35 – Lições aprendidas com a implantação de técnicas e ferramentas para detecção de vulnerabilidades

Lições aprendidas apontadas pelos respondentes	Quantidade %
Garantir a qualidade do software desenvolvido	13,33
Manter a equipe confortável com questões novas à sua realidade	6,67
Evitar a acomodação em relação à segurança da aplicação	6,67
Compartilhar o conhecimento entre membros	6,67
Branco	53,33
Não há implantação de técnicas e ferramentas para detecção de vulnerabilidades	13,33
Total	100%

Fonte: Lígia Santos, 2017

As lições apontadas por 13,33% dos gerentes estão relacionadas a garantia da qualidade do software desenvolvido. Para proporcionar a qualidade do software desenvolvido, a implantação de técnicas e ferramentas para detecção de vulnerabilidades no projeto é realizada. Sendo assim, visto que o software poderá ser considerado confiável, seguro e íntegro, a qualidade estará presente, pois as técnicas e ferramentas foram implantadas conforme as especificações do cliente e atenderá suas necessidades relacionadas a segurança.

Manter a equipe confortável com questões novas à sua realidade foi um lição aprendida por 6,67% dos gerentes. Evitar a acomodação em relação a segurança de software foi uma lição aprendida citada por 6,67% dos gerentes. Em métodos ágeis é recomendável que todos os membros tenham conhecimentos mesmo que mínimos em todos os eventos a serem realizados durante o ciclo ágil. Dessa forma, é necessário que o ambiente seja cooperativo e dinâmico, onde os membros tenham autonomia, eficiência e confiança na realização das atividades.

Já para 6,67% dos gerentes a lição aprendida está relacionada ao compartilhamento de conhecimento entre membros. Logo é recomendável que exista um ambiente colaborativo, em que os membros interajam entre si, apontando suas forças e fraquezas, a fim de difundir e

equilibrar o conhecimento. 53,33% dos gerentes não responderam quais as lições aprendidas com a implantação de técnicas e ferramentas.

Finalmente, 13,33% dos gerentes apontaram não implantar técnicas e ferramentas em seus projetos.