Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Automatização de Testes em Equipes Ágeis: Um Estudo Qualitativo Usando Teoria Fundamentada

Gabriella Mayara Tavares Alves

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software - Qualidade de Software

Patrícia Machado e Tiago Massoni
(Orientadores)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

A474a Alves, Gabriella Mayara Tavares.
"Automatização de testes em equipes ágeis : um estudo qualitativo usando teoria fundamentada" / Gabriella Mayara Tavares Alves. – Campina Grande, 2017.
92 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2017.
"Orientação: Profa. Dra. Patrícia Machado, Prof. Dr. Tiago Massoni".
Referências.

1. Engenharia de Software. 2. Qualidade de Software. 3. Práticas Eleitorais. I. Machado, Patrícia. II. Massoni, Tiago. III. Título.

CDU 004.41(043)

Resumo

Com o crescimento da utilização de práticas ágeis, as atividades de teste devem adaptar-se à agilidade na absorção e implementação antecipada dos requisitos. Com base nisso, normalmente a automação de testes de sistema para aplicações web, desktop e *mobile*, é amplamente utilizada e desenvolvida para melhorar a qualidade do software, permitindo executar mais testes de forma mais frequente, quando comparado com a execução manual. No entanto, o custo da manutenção dos scripts de testes automatizados é considerado alto e, normalmente, as equipes não possuem pessoas especializadas em automação de testes de sistema. Sendo assim, existem poucos relatos na literatura referentes às lacunas que impedem a utilização das vantagens oferecidas na automação de testes de sistema em sua plenitude no contexto de equipes que utilizam métodos ágeis. Desta forma, este trabalho, através de um estudo empírico com entrevistas semi-estruturadas e da Teoria Fundamentada, busca coletar e analisar dados acerca de práticas utilizadas em equipes ágeis na automação de testes de sistema para elencar práticas que indiquem o melhor momento para iniciar a criação dos scripts de automação de testes de sistema. Além disso, busca contribuir com a literatura e consequentemente, possuir uma base teórica para que propostas de melhorias sejam realizadas futuramente. Como resultado, foram identificadas práticas comuns de automação de teste de sistema utilizadas nas equipes de desenvolvimento, como: iniciar a criação dos scripts de teste automáticos após algumas execuções manuais dos casos de teste, e até o requisito funcional tornar-se estável; a criação dos scripts de teste automáticos são planejados para iniciar a partir das funcionalidades que possuem os casos de testes manuais executados no Sprint anterior; e gerenciamento das alterações solicitadas pelo cliente para replanejar caso a solicitação tenha impacto nas funcionalidades já implementadas. Para estruturar os resultados obtidos, utilizamos os princípios da Teoria Fundamentada através da análise das entrevistas realizadas para coleta de dados.

Abstract

With the increasing popularity of agile practices, test activities must adapt to agility in special test automation and anticipated implementation of requirements. Based on this, typically the automation of system tests for web, desktop and mobile applications is largely used to improve software quality, allowing for more frequent testing, when compared to manual execution. However, the maintainance cost of automated tests is high, and teams normally do not have specialized people in test automation. Therefore, the literature lacks reports related to the gaps that prevent the use of the advantages offered by the system testing automation in its fullness in agile teams. This work, through an empirical study with semi-structured interviews and the Fundamentated Theory, aims to collect and to analyze data about practices used in agile teams in the system testing automation to list practices that indicate the best moment to start the creation of the system test automation scripts. In addition, it seeks to contribute to the literature and consequently a theoretical basis, so that suggestions for improvements can be made in the future. The collected data allowed us to identify system testing automation practices used in typical agile teams, such as starting the creation of automated test scripts after some manual executions of the test cases, until the functional requirement becomes stable; the activities of creationing automatic test scripts should be planned to start from the features that have the manual test cases executed in the previous Sprint; and the management of the changes requested by the client to replanning quickly if the request causes a major impact on the features in validation status. To structure the obtained results, the principles of the Grounded Theory were used through the analysis of the interviews conducted for data collection.

Agradecimentos

Agradeço,

A Deus por sempre estar presente na minha vida, me guiando para os melhores caminhos e me dando força para enfrentar e superar as dificuldades da vida.

A Nossa Senhora, por sempre interceder à seu filho Jesus, o melhor para minha vida.

Aos meus pais, Arizonete e Marcos, por sempre estarem presente nessa caminhada, me incentivando e confiando nas minhas decisões.

A meu namorado, Túlio, pela compreensão e ajuda nos momentos necessários, além disso, pelo incentivo para a realização dos meus sonhos.

Aos meus orientadores, Professora Patrícia Machado e Professor Tiago Massoni, por serem pessoas extraordinária e humanas, fornecendo todo suporte ao longo dessa trajetória, com suas correções, direcionamentos, paciência e incentivo.

Aos meus colegas de trabalho, com quem compartilhei momentos de alegria, estresse e tensão.

Aos participantes da pesquisa, por voluntariamente ter contribuído com minha pesquisa.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

Conteúdo

1 Introdução **1**

1.1 Contexto 1

1.2 Problema 3

1.3 Trabalhos Relacionados 4

1.4 Solução 6

1.4.1 Objetivo Geral 6

1.4.2 Objetivos Específicos 6

1.4.3 Método de Pesquisa 6

1.5 Principais Resultados 7

1.6 Organização da Dissertação 8

2 Fundamentação Teórica **9**

2.1 Pesquisa Qualitativa 9

2.2 Teoria Fundamentada - TF 11

2.2.1 Confiabilidade entre os codificadores 13

2.3 Métodos Ágeis em Engenharia de Software 15

2.4 Teste de Software 17

2.4.1 Teste de Software em Ambientes Ágeis 21

2.5 Automação de Teste de Software 22

2.5.1 Limitações da Automação de Teste de Sistema 24

2.6 Considerações Finais do Capítulo 26

3 Metodologia do Estudo **27**

3.1 Perguntas de Pesquisa 27

3.2 Contexto do Estudo 28
3.3 Procedimento Experimental 29
3.4 Análise dos resultados 30
3.4.1 Seleção de fragmentos das entrevistas 31
3.4.2 Categorização dos fragmentos da entrevista 31
3.4.3 Agrupamento das categorias em áreas temáticas 33
3.4.4 Cálculo da confiabilidade 33
3.4.5 Pretensão de resposta as perguntas de pesquisa 34
3.5 Considerações Finais do Capítulo 35

4 Resultados do Estudo **36**
4.1 Comunicação 38
4.2 Documentação 40
4.3 Pré Requisito 42
4.4 Execução de testes 44
4.5 Automação de testes 46
4.6 Manutenção dos Testes 48
4.7 Considerações Finais do Capítulo 50

5 Análise dos Resultados **51**
5.1 PPS_01 - Quais são as dificuldades para automação de testes de sistema? . . 51
5.2 PPS_02 - Como é realizado o gerenciamento dos artefatos relacionados aos testes de sistema? 53
5.3 PPS_03 - Quais são as dificuldades em manter os testes de sistema? 54
5.4 PPS_04 - Como as mudanças afetam os scripts automatizados de teste de sistema? 55
5.5 PPS_05 - Como são executados os testes de sistema e quais outros testes são executados? 57
5.6 Pergunta Principal: Qual o momento mais adequado para iniciar a criação dos scripts automáticos dos testes de sistema? 58
5.7 Ameaças 59
5.8 Considerações Finais do Capítulo 60

6 Considerações Finais **62**

6.1 Conclusões 62

6.2 Trabalhos Futuros 65

A Termo de Consentimento **74**

B Roteiro da Entrevista Via E-mail **76**

C Roteiro da Entrevista Via Gravação **79**

D Arquivos Gerais **82**

Lista de Símbolos

TF - *Teoria Fundamentada*

PO - *Product Owner*

Lista de Figuras

2.1 Fases da Teoria Fundamentada 13
2.2 Modelo “V” (Fases do Desenvolvimento X Nível dos Testes) 18

3.1 Processo da Análise dos Dados 31

Lista de Tabelas

3.1 Exemplo de fragmento da entrevista 31

3.2 Exemplo da relação Categoria x Fragmento 32

3.3 Exemplo da relação área temática x categoria 33

4.1 Fragmentos por Área Temática - Categoria 37

4.2 Análise de confiabilidade 39

5.1 Cenário de desenvolvimento x Abordagem na automação de testes 58

Capítulo 1

Introdução

Neste capítulo é exposto o contexto que o trabalho de pesquisa estar inserido e os problemas que foram encontrados, no qual se busca resolver ao final do estudo. Apresentamos os trabalhos que estão relacionados com o contexto deste trabalho. Em seguida, apresentamos os objetivos principais, objetivo específicos e o método de pesquisa selecionado para abordar a problemática. Por fim, são apresentados um resumo dos principais resultados encontrados.

1.1 Contexto

Métodos ágeis de desenvolvimento de software têm evoluído em números de adeptos [3], principalmente pela filosofia de resultados mais rápidos para o usuário final, que permite resultados visíveis logo nos primeiros meses do projeto [25, 66]. Isso é atrativo para clientes interessados em ter seu sistema em uso o mais cedo possível. Entretanto, algumas características necessárias para aplicação dos métodos ágeis nem sempre são possíveis, como disponibilidade de unir em um mesmo espaço físico os desenvolvedores, dificuldade em trabalhar com grandes equipes, além de ter interferência do cliente com habilidades técnicas como por exemplo, representantes do cliente que tenham poder de decisão e ao mesmo tempo conhecimento profundo dos requisitos próximos aos desenvolvedores [39]. Além disso, a capacitação técnica e experiência dos desenvolvedores é de extrema importância para o sucesso na aplicação dos processos.

Para garantir que erros não ofusquem a boa impressão dos métodos ágeis no decorrer do projeto, os testes de software são considerados vitais ao processo. Assim, para a garantia da

qualidade do software, a atividade de teste de software é um elemento crítico e representa a última revisão de especificações, projeto e codificação. Perin [53] destaca que os custos envolvidos associados às falhas de software são critérios propulsores para uma atividade de teste cuidadosa e bem planejada. Logo, com a importância da atividade de teste, e o crescimento da utilização de práticas ágeis, os testes de software devem ocorrer de forma frequente, procurando detectar defeitos o mais cedo possível em ciclos de desenvolvimento iterativos e curtos, com um constante feedback tanto da equipe de teste, quanto para o cliente.

Sendo assim, a atividade de teste de software é normalmente realizada em diferentes níveis ao longo dos processos de desenvolvimento e manutenção. O foco desta dissertação é no nível de teste de sistema, para aplicações web, desktop e *mobile*, e assim os testadores especificam os cenários que tendem a cobrir o maior número de comportamentos esperados de cada funcionalidade do sistema, construindo artefatos como, por exemplo, os casos de testes, e assim, sua execução pode ser realizada automaticamente ou/e manualmente [50, 81]. Os testes manuais são construídos a partir da especificação do software e normalmente são fáceis de serem aplicados, uma vez que o usuário utiliza o sistema como se o mesmo estivesse concluído, além de ser intuitivo onde pessoa sem conhecimento técnico prévio, realiza a execução dos testes especificados [45]. Porém, o custo com a execução torna-se alto para o desenvolvimento do software por requerer muito tempo de um testador; um exemplo disso é uma execução de regressão.

Já com os testes de sistema automatizados, assim como os testes manuais, os scripts dos testes são criados através da especificação do software ou seguindo os casos de testes manuais, ou seja, sem acesso ao código que foi criado (chamado teste caixa preta), além de serem executados a qualquer momento e com pouco esforço [8]. Conforme Maldonado [47], a utilização de ferramentas para criação de scripts de testes automatizados de sistema em aplicações web (por exemplo, Selenium WebDrive[1] e Robot Framework[2]) é essencial para que os testes tenham um alto grau de confiabilidade, contribuindo expressivamente para a redução do custo de tempo na criação e execução dos scripts de testes [67]. No entanto, testes de sistema podem incluir acesso a interface com o usuário, cujas ferramentas para automação são difíceis de usar, talvez por isso há pouca automatização de teste.

[1]Disponível em: http://www.seleniumhq.org/

[2]Disponível em: http://robotframework.org/

Geralmente, a especificação existente para a criação dos testes é informal e, desse modo, a determinação da cobertura obtida por um conjunto de casos de testes ou scripts dos testes automatizados também é informal [53]. Diante disso, testes de sistema em ambientes ágeis precisam lidar frequentemente com pouca ou nenhuma documentação de requisitos.

1.2 Problema

Como os casos de testes manuais são importantes na garantia da qualidade do sistema, geralmente considera-se que sua especificação de alto nível por ser construída em linguagem natural. Logo, a mudança e manutenção desses casos são realizadas frequentemente, por serem mais simples de atualizar. Por outro lado, os scripts dos testes automatizados necessitam de pessoas especializadas e maior tempo para realizar essas alterações, por consequência dos scripts dos testes automatizados serem dedicados a uma versão do software, sendo sua manutenção menos frequente. Portanto, Fewster [24] afirma que, se a manutenção dos scripts de testes for ignorada, a atualização de toda a suite de testes automatizados tem um custo maior do que a atualização dos testes manuais. Porém, de acordo com Dustin [22], o aumento da eficiência e o alto custo relacionado ao tempo para a execução dos testes manuais, especialmente em testes de regressão, em que os casos de teste são executados de forma iterativa em cada versão do software, são razões para automatizar os testes de sistema.

Alégrotha [1] apresenta em seu estudo que o custo da manutenção dos scripts de testes automatizados está relacionado a 13 fatores, como por exemplo, o conhecimento/experiência do testador. Diante disso, como as equipes estão em busca do menor tempo para execução dos scripts e possuem demanda por qualidade do sistema, é complexo definir o melhor momento no processo de teste que os scripts dos testes automatizados no nível de sistema devem ser criados, e assim, utilizando-os de maneira produtiva. Uma vez que, a decisão por automatizar no início do desenvolvimento das funcionalidades, o risco desses scripts futuramente passarem por grandes alterações ou até mesmo serem descartados pode ser alto, influenciando no cronograma do desenvolvimento. Contudo, deixar para realizar a automação desses testes após o desenvolvimento das funcionalidades, pode aumentar o risco de exceder o custo x benefício, além da obsolescência dos testes automatizados passa a existir; o custo com alterações pode até diminuir, mas em contraposto, se automatizar tarde demais, pode-se

aumentar o custo de tempo e esforço do processo de testes.

Logo, para usufruir das vantagens dos scripts de testes automatizados e dos problemas em não automatizar cedo demais ou tarde demais os scripts de testes, levanta-se um questionamento. Qual o momento durante o desenvolvimento do software ideal para iniciar a criação dos scripts de testes automatizados? Diante desse questionamento, existem poucos relatos na literatura sobre o momento que a automação é iniciada no processo de desenvolvimento ágil. Geralmente encontra-se relatos de definição de processos utilizados e identificação das habilidades dos profissionais por exemplo [40, 56, 84], mas nada sobre o início da automação dos testes de sistema, sem que ocorra muitas alterações nos scripts dos testes automatizados criados e a equipe obtenha as vantagens da automação. Portanto, estamos buscando evidências de como se faz a criação dos scripts da prática, para obter conhecimento da diversidade dos processos de desenvolvimento ágeis, assim como das práticas utilizadas por cada um deles.

1.3 Trabalhos Relacionados

Serão apresentados alguns trabalhos que refletem o estado da arte no contexto de estudo qualitativo na criação de scripts de testes automático em ambiente de desenvolvimento ágil. Buscamos, quando possível, localizar lacunas existentes, verificar os objetivos para direcionar o trabalho desenvolvido e comparações dos mesmos com o trabalho apresentado neste documento.

A abordagem qualitativa tem sido frequentemente utilizada em estudos voltados para a compreensão da vida humana em grupos, na sociologia, antropologia, psicologia, ciências sociais e saúde [59]. Sendo assim, Cresswell [17] destaca a teoria fundamentada como uma das modalidades inerentes nas pesquisas humanas e sociais. Não era habitual ser encontrado na literatura pesquisas com análise qualitativa utilizando teoria fundamentada na engenharia de software com métodos ágeis, no entanto, nos últimos anos, tem havido um crescimento no número de pesquisadores explorando aspectos humanos e sociais da Engenharia de Software e com isso utilizando a Teoria Fundamentada [36] Por exemplo, o trabalho de Chakraborty et al. [13], Crabtree et al. [16], Hoda et al. [36], Parizi et al. [52] e Stojanov et al. [71], que possuem o objetivo de abordar problemáticas da Engenharia de Requisitos de Software,

Métodos ágeis, Processo de Software e Implementação de Sistemas.

Os trabalhos encontrados que possuem foco na Qualidade de software, foram destacados através do estudo de Greiler et al. [34], Kasurinen et al. [40], Prechelt et al. [56], Taipale et al. [74] e Yu et al. [84]. O estudo de Greiler et al. [34] procurou compreender o que pensam desenvolvedores e testadores ao validar sistemas que dependem de plug-ins com uma análise qualitativa, porém foi realizada uma etapa quantitativa para validar os resultados da fase qualitativa; o estudo de Kasurinen et al. [40] procurou identificar problemas em equipes de testes e relacionar a existência de tais problemas com o tamanho da organização utilizando Teoria Fundamentada; o estudo de Prechelt et al. [56] investigou o processo e detalhou as vantagens e desvantagens da utilização de equipes de testes ágeis através da teoria fundamentada; o estudo de [74] buscou entender a prática de testes de software para derivar hipóteses sobre as organizações de teste e gestão do conhecimento através da Teoria Fundamentada; e o estudo de Yu et al. [84], procurou compreender as habilidades "*soft*" de engenheiros de testes e sua influência em projetos, porém o trabalho não apresenta nenhuma evidência do uso da Teoria Fundamentada como conceitos ou categorias identificados. Porém, esses trabalhos enfatizam os problemas relacionados ao processo das atividades da equipe de qualidade, não existindo um foco na problemática quando relacionada a automação de teste em ambientes ágeis. Outro ponto que não foi abordado nesses trabalhos se refere a falta de informação do tipo de metodologia utilizada no processo de desenvolvimento, com exceção de Prechelt et al. [56].

A dificuldade na busca de trabalhos adotando pesquisa qualitativa para retratar as problemáticas em automação de teste de sistema no ambiente de desenvolvimento ágil esteva presente na seleção dos trabalhos. Entretanto, foi destacado alguns trabalhos que se aproximam do objetivo do estudo apresentado neste trabalho, por exemplo, Deak and Stlhane [21], Kasurinen et al. [41], Taipale et al. [75] e Tyagi et al. [79]. O estudo de Deak and Stlhane [21] identificou os desafios diários encontrados no mundo real das organizações de software norueguesas e discutiu as dificuldades enfrentadas pelos testadores em suas atividades, além disso observou as possíveis conexões entre a automação de atividades de teste de software e a organização de testes; o estudo de Kasurinen et al. [41] observou e identificou fatores que afetam os testes, com a automação como aspecto central em diferentes tipos de organizações; o estudo de Taipale et al. [75] analisou o uso de automação de teste em organizações

de teste de software para identificar o que facilita o uso da automação de teste em uma organização e as dificuldades do uso de automação de teste; e o estudo de Tyagi et al. [79] buscou criar um entendimento sobre os diferentes desafios enfrentados pelos profissionais ágeis, adotando a automação de testes em projetos ágeis e apresentou algumas estratégias possíveis para superar esses desafios, utilizando Teoria fundamentada. Embora esses trabalhos retratem um pouco do cenário da automação em ambientes ágeis, os autores não citam qual o nível de automação de testes que buscaram investigar em seu estudo, portanto não é confiável destacar os resultados encontrados por eles com o objetivo desta pesquisa, uma vez que o nosso objetivo é investigar os cenários de desenvolvimento que utilizam automação de teste de sistema em ambientes de desenvolvimento ágil.

1.4 Solução

1.4.1 Objetivo Geral

O objetivo geral deste trabalho é, através de um estudo empírico, **coletar** e **analisar** dados acerca das práticas utilizadas em equipes ágeis para a automação de teste de sistema.

1.4.2 Objetivos Específicos

- Coletar evidências através de entrevistas semiestruturadas com profissionais da área de tecnologia e integrados em ambiente de desenvolvimento de projeto real utilizando metodologias ágeis.
- Analisar qualitativamente os dados coletados, construindo conclusões através de a Teoria Fundamentada [30, 72].

1.4.3 Método de Pesquisa

O presente trabalho de pesquisa se encaixa no perfil do estudo de caso, cujo objeto é uma unidade que se analisa profundamente, e por sua vez, tem se tornado uma estratégia preferida quando os pesquisadores procuram responder questões 'Como?' e 'Por quê?' certos fenômenos ocorrem, quando há pouca possibilidade de controle sobre os eventos estudados

e quando o foco de interesse é sobre fenômenos atuais, que só poderão ser analisados dentro de algum contexto de vida real [31].

Este método é útil quando o fenômeno a ser estudado é amplo e complexo e não pode ser estudado fora do contexto onde ocorre naturalmente, buscando determinar ou testar uma teoria, e tem como uma das fontes de informações mais importantes, as entrevistas. Através delas, o entrevistado expressa sua opinião sobre determinado assunto, utilizando suas próprias interpretações.

Por isso, neste estudo, a coleta de dados foi realizada através de entrevistas de maneira presencial e via e-mail. Após a conclusão de cada entrevista, iniciamos o processo de análise dos dados através da Teoria Fundamentada [30, 72], no qual periodicamente, durante a análise dos dados, necessitamos entrar em contato com o entrevistado novamente para esclarecimento de dúvidas. Para tal, foram contactadas 40 pessoas para participar da pesquisa, entre gerente de projetos e líderes de qualidade que trabalham em grandes empresas nacionais e internacionais, através de listas de discussões sobre qualidade de software e colegas que atuam na área de qualidade de software. Porém, apenas 25 participantes confirmaram a participação através da resposta do termo de consentimento. Dentre estes, 52% optaram por realizar entrevista presencialmente, com áudio gravado, enquanto que 48% optaram por entrevista via e-mail. Dentre os que optaram por e-mail, 33.3% não responderam o e-mail com as perguntas da entrevistas. Portanto, no total tivemos 21 participantes, sendo 13 por entrevista presencial e 8 por entrevista via e-mail. Seguindo a Teoria Fundamentada, adotamos a seleção dos fragmentos de cada entrevista de acordo com as perguntas de pesquisa, consequentemente a categorização desses fragmentos e definição das áreas de temática.

1.5 Principais Resultados

Com a avaliação dos dados, obtivemos informações sobre as práticas na automação de teste de sistema em equipes ágeis. Observamos que uma prática comum é iniciar a criação dos scripts apenas quando o requisito está estável, ou seja, não existem solicitações de grandes mudanças; e após a execução dos testes manuais, o que geralmente produz scripts mais estáveis. Identificamos também que equipes que possuem requisitos bem definidos, e pessoas especializadas em automação de teste de sistema, iniciam a criação dos scripts dos testes au-

tomatizados de sistema posterior à primeira execução dos testes manuais. Observamos que equipes cujos membros adotam papel de desenvolvedor e testador em momentos diferentes não apresentam definição do momento para iniciar a automação de testes de sistema, uma vez que normalmente esses profissionais tem que dividir seu tempo para implementar e validar as funcionalidade e não possuem conhecimento técnico nos *frameworks* e práticas de automação de testes de sistemas. No entanto, tivemos uma pequena participação de equipes que implementam muitas mudanças de requisitos e não utilizam nenhum tipo de automação de testes de sistema em seu processo de qualidade, uma vez que existe o déficit de profissionais com experiência em automação de testes de sistema.

1.6 Organização da Dissertação

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2, apresentamos informações básicas sobre os principais conceitos utilizados nesta dissertação. Já no Capítulo 3 apresentamos a estratégia utilizada como metodologia para coletar e analisar os dados. Portanto, no Capítulo 4, apresentamos os resultados analisados a partir da Teoria Fundamentada. O Capítulo 5 detalha a análise dos dados com base nas perguntas de pesquisa. Finalmente no Capítulo 6, apresentamos as conclusões, os trabalhos relacionados e as limitações deste trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo é exposto uma visão geral dos conceitos importantes que servem como base para fundamentar esta pesquisa. Inicia-se com a definição do objetivo da pesquisa qualitativa. Em seguida, apresenta-se a teoria fundamentada, a qual é uma metodologia utilizada nas pesquisas qualitativas. Posteriormente, uma breve apresentação da análise de conteúdo, a confiabilidade entre codificadores, seguida de uma apresentação sobre processos ágeis. Por fim, são apresentadas definições de algumas fases de testes de softwares, testes de softwares em ambientes ágeis, assim como definições da automação de teste com suas vantagens e limitações.

2.1 Pesquisa Qualitativa

A pesquisa qualitativa não se preocupa com representatividade numérica, mas, sim, com o aprofundamento da compreensão de um grupo social, de uma organização, etc. Os pesquisadores que adotam a abordagem qualitativa opõem-se ao pressuposto que defende um modelo único de pesquisa para todas as ciências, já que as ciências sociais têm sua especificidade, o que pressupõe uma metodologia própria. Portanto, assume diversas formas, através das questões amplas que vão se esclarecendo no decorrer da pesquisa a ser conduzida através de diferentes caminhos [63, 69, 78]. Logo, utilizamos o estudo de caso, cujo objeto é uma unidade de que se analisa profundamente.

Segundo Ludke [44] o estudo de caso possui sete características essenciais, são elas:

i. Buscar descobertas mesmo que o pesquisador parta de alguns pressupostos teóricos

iniciais, teoria que servirá de esqueleto ou estrutura básica a partir da qual novos aspectos poderão ser detectados;

ii. Ênfase na interpretação em contexto para compreender melhor a manifestação geral de um problema, as ações, percepções, comportamentos e interações das pessoas devem ser relacionadas à situação específica onde ocorrem, ou à problemática determinada a que estão ligadas;

iii. Retratar a realidade de forma completa e profunda, procura revelar a multiplicidade de dimensões presentes numa determinada situação ou problema;

iv. Experiência vicária e permite generalizações naturalísticas, o pesquisador procura relatar as suas experiências durante o estudo de modo que o leitor ou usuário possa fazer as suas generalizações naturalísticas;

v. Representar os diferentes, às vezes, também conflitantes pontos de vista presentes numa situação social;

vi. Utilizar uma linguagem e uma forma mais acessível do que os outros relatórios de pesquisa.

Além disso, Ludke [44] destaca que existem três fases no desenvolvimento de um estudo de caso: uma primeira aberta ou exploratória, uma segunda mais sistemática em termos de coleta de dados e uma terceira, com a análise e interpretação dos dados, seguidas da elaboração de um relatório.

Com isso, para a presente pesquisa, os procedimentos metodológicos foram realizados a partir da coleta de dados através do contato direto e interativo do pesquisador com profissionais da área de qualidade de software. Dessa maneira, essa pesquisa qualitativa teve como foco entender e interpretar as informações presentes nas entrevistas. Então, essa metodologia repousa sobre a interpretação dos dados com a utilização de várias técnicas de análise de discurso, como a Teoria Fundamentada [70] e o Recal3 [19].

2.2 Teoria Fundamentada - TF

Seguindo os princípios da metodologia qualitativa, a teoria fundamentada é uma metodologia de campo que objetiva gerar construtos teóricos que explicam ação no contexto social sob o estudo [12]. Além disso, a análise e o conceito são gerados através de processos intercalados de coleta e análise de dados utilizando a comparação constante. Sendo assim, os códigos são selecionados por um processo de amostragem teórica [80]. Para representar o método de abordagem, optamos por reunir os passos de Stern [70], Glaser e Strauss [30], e Strauss e Corbin [72], no qual apresentam: coleta de dados empíricos, procedimentos de codificação e análise dos dados (codificação aberta, codificação axial e codificação seletiva) e definição da teoria.

Em resumo, de acordo com Strauss e Corbin [73], os procedimentos de codificação envolvem: i) construir em vez de testar a teoria; ii) fornecer aos pesquisadores ferramentas analíticas para lidar com as massas de dados brutos; iii) ajudar os analistas a considerar significados alternativos para os fenômenos; iv) ser sistemático e criativo simultaneamente; e v) identificar, desenvolver e relacionar os conceitos que são os blocos de construção da teoria. Os processos de codificação são denominados de codificação aberta, axial e seletiva [29], como formas diferentes de tratar os dados, muito mais do que etapas firmemente demarcadas, claramente distintas e temporalmente separadas.

Na Codificação Aberta, os pesquisadores analisam os dados manualmente, submetendo-os ao processo de codificação linha a linha, no qual, são formados fragmentos textuais [1] que expressam a essência do discurso dos depoentes e atribuindo palavras ou expressões para esses fragmentos, formando então os códigos preliminares. Ou seja, é nesta fase da codificação que os dados são codificados, comparados com outros dados e designados em categorias. Através da codificação aberta, o pesquisador deve fazer comparações e perguntas para guiá-lo no campo empírico como, por exemplo: O que está acontecendo? Em quais categorias esses dados se enquadram? O que os dados expressam? [27]. Por isso que Cassiani [12], expõe que o pesquisador codifica os incidentes em muitas categorias, pois neste momento todos os dados são passíveis de uma codificação.

[1] Fragmento textual é caracterizado pela concentração em um determinado tema, com extensões variadas, que vão desde o âmbito do enunciado, correspondendo aproximadamente ao conceito de período, do ponto de vista sintático, até um âmbito mais abrangente envolvendo porções maiores do texto [38].

A codificação Axial é responsável por aprimorar e diferenciar as categorias resultantes da codificação aberta. O pesquisador, dessa forma, seleciona as categorias mais relevantes e as coloca como fenômeno central para estabelecer relações entre as categorias e subcategorias. Os dados, portanto, são agrupados através das conexões entre as categorias [54]. Com isso, após esse processo indutivo de agrupamento dos códigos em categorias, os dados são analisados comparativamente, à luz dos novos dados que estão chegando, com o intuito de tentar identificar as categorias mais significativas [12].

A codificação seletiva objetiva refinar e integrar categorias, classificando uma categoria que se considere como central, além da organização adequada dos códigos, categorias e subcategorias emergidas, assim como a evidência da categoria central que nasce mediante a relação dos agrupamentos. "O fenômeno central é o coração do processo de integração"[72] As condições causais, o contexto, as condições intervenientes, as estratégias e consequências formam as relações teóricas pelas quais as categorias são relacionadas uma a outra e à categoria central. Esse procedimento, força o investigador a desenvolver alguma estrutura teórica e é denominando paradigma de análise. O paradigma se constitui, portanto, do seguinte formato: (i) Condições causais => Fenômeno => Contexto => Condições; e (ii) Intervenientes => Estratégias de ação/interação => Consequências [12]. Com isso, após a codificação passa-se para a fase seguinte, a construção da teoria.

Para a definição da teoria, Strauss e Corbin [73] afirmam que a TF tende a se parecer mais com a realidade do que a teoria derivada da reunião de uma série de conceitos baseados em experiência ou somente por meio de especulação, em que se raciocina sobre como algo deveria funcionar. Para eles, o valor da TF está em sua capacidade não apenas de gerar teoria, mas de basear essa teoria em dados. Teorias fundamentadas, por serem baseadas em dados, tendem a oferecer mais discernimento, melhorar o entendimento e fornecer um guia importante para a ação [76]. Para construir uma teoria baseada na TF deve-se ter em vista dois procedimentos: desenvolvimento de uma história analítica clara por meio dos diagramas e memorandos e o delineamento de um esquema principal provisório que incorporará os componentes relevantes da história [72]. Os esclarecimentos dos conceitos e das principais linhas de desenvolvimento auxiliam muito na construção de uma teoria concisa.

Portanto, as várias fases da Teoria Fundamentada ocorrem simultaneamente, permitindo ao pesquisador fazer as modificações necessárias no transcorrer do processo (Figura 2.1).

Figura 2.1: Fases da Teoria Fundamentada

O procedimento de retroalimentação constante com os indivíduos da pesquisa possibilita entender melhor o fenômeno estudado. Assim, à medida que os dados são coletados e analisados, surgem outros novos que direcionarão as novas coletas, produzindo categorias mais refinadas até a saturação [29].

2.2.1 Confiabilidade entre os codificadores

Para Swert [20], a confiabilidade entre codificadores é crucial para pesquisas que realizam análise de conteúdo. No entanto, não parece haver um padrão geral sobre como fazer e relatar os testes de confiabilidade entre codificadores. Nas publicações acadêmicas, são apresentadas diversas medidas. As razões para essa falta de uniformidade é a falta de informações suficientes sobre o teste de confiabilidade entre codificadores, como esse teste é calculado e como os resultados desse teste deve ser interpretados.

A confiabilidade entre os codificadores é o termo amplamente utilizado nos estudos no qual os codificadores independentes avaliam uma característica de uma mensagem ou artefato e alcançam à mesma conclusão [32, 77]. Gore et al. [32] e Tinsly et al. [77] também relatam que, embora a confiabilidade possa ser baseada em índices correlacionais (ou de análise de variância) no qual avaliam o grau de confiabilidade, o acordo de entre codificadores é necessário na análise de conteúdo, mesmo quando o acordo entre codificadores é usado

para variáveis no intervalo ou relação com níveis de medição do acordo real sobre os valores codificados (mesmo que os valores similares, em vez de idênticos, sejam a base para a avaliação).

Como utilizado na medida com os codificadores independentes, a confiabilidade entre codificadores é um componente crítico da análise de conteúdo, uma vez que, quando não estabelecida adequadamente, os dados e suas interpretações não podem ser considerados válidos. Neuendorf [51] observa que um objetivo da análise de conteúdo é identificar e registrar características relativamente objetivas (ou pelo menos intersubjetivas) das mensagens. Assim, Kolbe e Burnett [42] defendem que a confiabilidade entre juízes é muitas vezes percebida como a medida padrão da qualidade da pesquisa. As altas taxas de desacordo entre juízes sugerem falhas nos métodos de pesquisa, incluindo a possibilidade de definições de fracas operações, categorias e treinamento.

Existem importantes razões práticas para estabelecer a confiabilidade entre codificadores. Neuendorf [51] expõe que, além de ser uma condição necessária, embora não suficiente, é mais uma etapa na validação do esquema de codificação, estabelecendo um alto nível de confiabilidade. Para Rust e Cooil [58], a concordância entre a confiabilidade é importante para o marketing dos pesquisadores ao destacar que a "alta confiabilidade torna menos provável que más decisões gerenciais resultem no uso dos dados". Resumindo, Lombard [43] expõe que os pesquisadores de análise de conteúdo devem se preocupar com a confiabilidade entre codificadores porque com sua avaliação adequada pode tornar a codificação mais eficiente, entretanto se não se preocupar com a confiabilidade entre codificadores, muito provavelmente sua pesquisa será rejeitada por críticos.

Existem vários índices de concordância entre codificadores Popping [55] identificou 39 "índices de concordância" diferentes para a codificação categorias nominais, o que exclui diversas técnicas para dados de nível de intervalo e de relação, mas apenas algumas técnicas são amplamente utilizadas. No entanto, foram selecionados dois dos coeficientes de confiabilidade mais populares para dados nominais: concordância percentual e coeficiente de Kappa de Cohen.

A concordância percentual parece ser mais utilizada e intuitivamente atraente e simples de calcular, além disso é uma medida de concordância entre vários codificadores para a forma como eles aplicam códigos aos dados de texto [37]. Já a estatística de kappa tem

sido recomendada como uma medida útil para quantificar o acordo de codificadores [10]. A confiabilidade foi aferida pelo percentual de concordância simples entre os dois codificadores e através da estatística kappa de Cohen. O valor do coeficiente kappa é obtido a partir da fórmula: k = Po - Pe/(1 - Pe), onde Po = proporção global de concordância observada e Pe = proporção global de concordância esperada ao acaso. Assim, kappa é interpretado como a proporção de acordo possível além do acaso que foi realmente alcançado.

Para calcular a concordância percentual e o coeficiente Kappa de Cohen, utilizamos na atual pesquisa o software Recal [28] que está disponível, juntamente com a documentação e as instruções, na página web gratuitamente[2]. Recal ("Calculadora de Confiabilidade") é um utilitário online que calcula entre múltiplos coeficientes de confiabilidade entre a nominal, ordinal, intervalo, ou em nível de relação de dados.

Desse modo, ele consiste em três módulos independentes cada um especializado para diferentes tipos de dados, que são definidos como: (i) nominal com apenas 2 codificadores – Recal2; (ii) nominal com 3 ou mais codificadores – Recal 3; e (iii) ordinal, intervalo ou razão com N de codificadores – Recal Oir[3]. Pelo fato de utilizar o Recal para calcular a confiabilidade da análise realizada com base nas categorias criadas, percebeu-se a necessidade da utilização do Recal3 por possuir mais de 3 categorias na análise de cada pergunta de pesquisa.

Desta forma, pensamos utilizar os princípios da Teoria Fundamentada com o acréscimo dos cálculos para a confiabilidade entre os codificadores em ambientes de práticos que utilizam métodos ágeis no seu processo de desenvolvimento.

2.3 Métodos Ágeis em Engenharia de Software

As "Metodologias Ágeis" tornaram-se populares em 2001 quando especialistas em processos de desenvolvimento de software representando os métodos Scrum [62], Extreme Programming (XP) [7] e outros, estabeleceram princípios comuns e compartilhados por todos esses métodos. Foi então criada a Aliança Ágil e o estabelecimento do "Manifesto Ágil" [48]. Com isso, surgiram definições para classificar a utilização da metodologia ágil, alguns des-

[2]Disponível em: http://dfreelon.org/utils/recalfront/

[3]Disponível em: http://dfreelon.org/utils/recalfront/

ses conceitos são [65]:

- Indivíduos e interações ao invés de processos e ferramentas;
- Software executável ao invés de documentação;
- Colaboração do cliente ao invés de negociação de contratos;
- Respostas rápidas a mudanças ao invés de seguir planos.

Os processos ágeis seguem uma lógica empírica num ciclo PDCA (*Plan-Do-Check-Act*), onde algum desenvolvimento é planejado e feito, os resultados são inspecionados e são feitas adaptações para melhorar o processo para resolver quaisquer problemas que tenham surgidos [83]. Uma vez que, atualmente o desenvolvimento de software é conduzido em ambientes de negócios cada vez mais turbulentos, tipicamente, os mercados de rápida mudança e imprevisíveis, as exigências complexas e mutáveis dos clientes, as pressões de menor tempo de colocação no mercado e as tecnologias de informação que avançam rapidamente são características encontradas na maioria dos projetos de desenvolvimento de software. Para enfrentar esta situação, as práticas ágeis [14] defendem flexibilidade, eficiência e rapidez são vistas como cada vez mais atraentes pelas empresas de desenvolvimento de software.

Sendo assim, esses conceitos estão relacionados com constantes alterações no sistema. Por consequência, os requisitos são passíveis de mudanças, as equipes são pequenas e o processo de desenvolvimento mais rápido, por possuir datas curtas de entrega do software. Além disso, as equipes de desenvolvimento assumem uma maior responsabilidade na organização do trabalho, passando a se reunir periodicamente de forma a partilhar detalhes do projeto.

Um dos mais populares e amplamente adotado, é o método ágil Scrum. O ciclo de vida de um projeto Scrum é em grande parte composto de iterações com desenvolvimento de "Sprints", com uma fase de planejamento inicial e uma fase final de encerramento com uma revisão e retrospectiva do Sprint. O planejamento permite abordar as questões arquitetônicas e de escopo e a fase de encerramento incorpora a gestão de *releases*. As fases inicial e final são sugeridas como previsíveis e podem ser definidas. A natureza empírica do Scrum é evidente nas iterações em andamento ou Sprints que se adaptam ao feedback e à mudança ao longo do projeto. Um aspecto notável desta abordagem é a inclusão de grupos que foram considerados obstáculos a projetos de desenvolvimento tradicionais (por exemplo, vendas e

marketing). O método abraça a mudança permitindo que a equipe de desenvolvimento reaja e promova mudanças à medida que o sistema evolui [25] e define os seguintes papéis na equipe: (i) *Scrum Master*, garante que as práticas e regras do Scrum sejam implementadas e seguidas, e resolva quaisquer problemas de recursos ou outros impedimentos que possam impedir a equipe de seguir as práticas e regras. Esta pessoa não é o líder da equipe, mas um treinador; (ii) *Product Owner*, representa o cliente gerando, mantendo e priorizando o backlog do produto. Esta pessoa não é a liderança da equipe; e (iii) Equipe de Desenvolvimento, desenvolve e testa o produto, no qual é auto-organizada [68].

No geral, o desenvolvimento ágil se refere a um grupo de metodologias de desenvolvimento de software baseadas em desenvolvimento interativo, onde os requisitos e soluções evoluem através da colaboração entre as equipes multifuncionais. Assim, idealmente, espera-se que todos os membros da equipe estejam aptos a realizar todas as tarefas. Na prática, normalmente existe algum grau de especialização entre os membros da equipe, e a maioria das equipes ágeis não atinge a meta de ser totalmente multifuncional, até porque as pessoas dificilmente gostam de executar todas as tarefas. Visando minimizar esse problema algumas empresas acabam criando, dentro do processo ágil, uma iteração para testes e outra para desenvolvimento [35].

2.4 Teste de Software

O Teste de software é considerado o elemento crítico da garantia de qualidade de software e deve ser conduzido por uma equipe de profissionais especializados. No entanto, um dos maiores problemas da indústria de software é a impassibilidade de desenvolver softwares sem erros. As empresas estão regularmente a obter relatórios de falhas nos seus produtos e este número aumenta a cada dia [64]. Perante um software cujo funcionamento não está de acordo com o que foi especificado, seja devido à existência de um erro ou à falha na sua implementação, o seu utilizador adquire um nível de insatisfação bastante considerável. Este mesmo desagrado também é partilhado por todos os intervenientes no processo de desenvolvimento de software [82].

De modo a reduzir o número de falhas, as empresas que desenvolvem software testam os seus produtos [82], simulando a sua utilização numa perspectiva de utilizador e de forma

a identificar, à partida, os problemas que possam existir. Assim encontrados, os problemas poderão ser corrigidos de modo a garantir uma maior qualidade do software. Portanto, seguindo alguns princípios, o teste de software pode tornar-se mais efetivo e assegura que os objetivos são cumpridos, apesar de existirem certas limitações. Testar software é importante para garantir a qualidade do mesmo e as atividades envolvidas necessitam ser bem planejadas, tendo em conta os seus objetivos, princípios e limitações que são mencionados a seguir.

Para atingir os objetivos do teste de software, é preciso definir um processo para estabelecer confiança de que o programa faz o que é suposto fazer, e para isso existem atividades de planejamento, especificação e execução. As atividades de testes de software, segundo Vieira [81], incluem planejar e controlar, escolher as condições de testes, desenvolver e executar os casos de teste, verificar os resultados, avaliar critérios de saída, manter os envolvidos informados sobre o decorrer do processo de testes, e por fim completar as atividades de encerramento, que são a elaboração de documentos com status de planejamento, escrita dos testes e status de execuções. Os testes também podem incluir a revisão de documentos e a realização de análise estática.

Contudo, além das atividades de teste, o processo de teste também possui níveis distintos os quais o software deve ser submetido, são eles: teste de aceitação, teste de sistema, teste de integração, teste de módulos e teste unitário. A Figura 2.2 apresenta a relação de cada nível de teste com a fase de desenvolvimento [4].

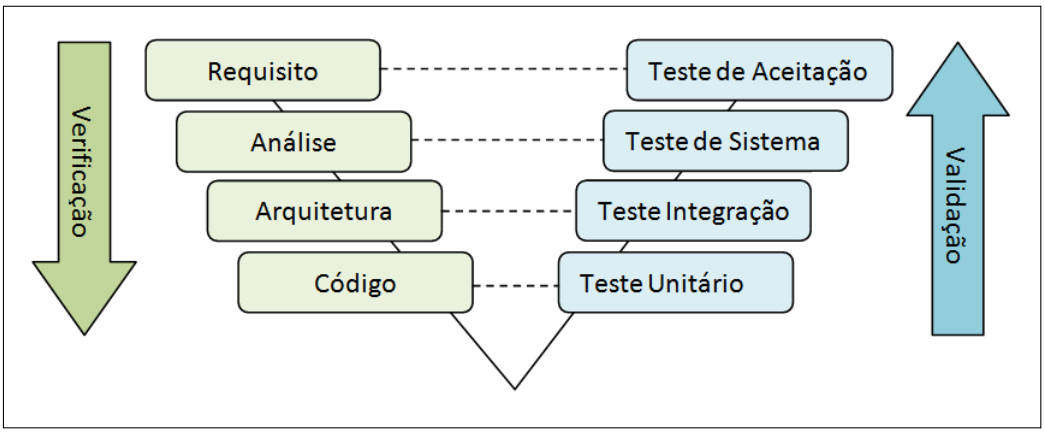

Figura 2.2: Modelo “V” (Fases do Desenvolvimento X Nível dos Testes)

Com isso, é possível se concentrar em testar o software sob vários aspectos de seu desenvolvimento, podendo detectar diversos tipos de falhas de acordo com a técnica empregada

em cada fase por possuir atividade de desenvolvimento de software distinta, ou seja:

- **Teste de Aceitação**: Testes formais conduzidos para determinar se um sistema satisfaz ou não os seus critérios de aceitação e para permitir ao cliente determinar se aceita ou não o sistema [57]. O teste de aceite pode ser realizado em mais de um único nível de teste, por exemplo [50]:
  - Um pacote de software ter um teste de aceite quando é instalado ou integrado;
  - Teste de aceite de usabilidade de um componente pode ser feito durante o teste de componente;
  - Teste de aceite de uma nova funcionalidade pode vir antes do teste de sistema.
- **Teste de Sistema:** Teste que verifica o desempenho, o comportamento e as funcionalidades do sistema estão conforme a especificação [9]. Teste de sistema se refere ao comportamento de todo do sistema, produto definido pelo escopo de um projeto ou programa de desenvolvimento. O ambiente de teste para execução deve corresponder o máximo possível ao objetivo final, ou o ambiente de produção, para minimizar que os riscos de falhas específicas de ambiente não serem encontradas durante o teste. Teste de sistema em requisitos funcionais deve inicialmente utilizar a técnica baseada em especificação mais apropriada (caixa-preta) de acordo com a característica do sistema a ser testado. Por exemplo, uma tabela de decisão pode ser criada por combinações de efeitos descritos em regras de negócio. A seguir, técnica baseada na estrutura (caixa-branca) pode ser utilizada para avaliar a eficácia do teste com respeito ao elemento estrutural, assim como estrutura do menu ou página web [50].
- **Teste de Integração:** Teste em que componentes de software, componentes de hardware ou ambos são combinados e testados para avaliar a interação entre eles [57]. É caracterizado por testar as interfaces entre os componentes, interações de diferentes partes de um sistema, como o sistema operacional, arquivos, hardware ou interfaces entre os sistemas. Pode haver mais de um nível de teste de integração, que pode ser utilizado em objetos de teste de tamanho variado. Por exemplo:
  - Teste de integração de componente testa interações entre componentes de software e é realizado após o teste de componente;

  - Teste de integração de sistemas testa interação entre diferentes sistemas e pode ser realizado após o teste de sistema. Neste caso a área de desenvolvimento pode controlar apenas um lado da interface, de forma que mudanças podem causar instabilidades. Processos de negócios implementados como fluxogramas podem envolver uma série de sistemas. Problemas relacionados a múltiplas plataformas podem ser significativos.

- **Teste de Componentes:** Teste que procura defeitos e verifica o funcionamento do software (ex.: módulos, programas, objetos, classes, etc.) que são testáveis separadamente. Pode ser feito isolado do resto do sistema, dependendo do contexto do ciclo de desenvolvimento e do sistema. Controladores ("*drivers*") e simuladores ("*stubs*") podem ser usados. Pode incluir teste de funcionalidade e características específicas não funcionais tais como comportamento dos recursos (ex.: falta de memória) e testes de robustez, além de teste estrutural (cobertura de código). Casos de teste são derivados dos produtos de trabalho como, por exemplo, especificação de componente, modelagem do software ou modelo de dados. Uma abordagem no teste de componente consiste em preparar e automatizar os casos de testes antes de codificar. Isto é chamado de abordagem de teste antecipado ou desenvolvimento dirigido a teste. Esta abordagem é essencialmente iterativa e é baseada em ciclos de elaboração de casos de testes. À medida que são construídas e integradas pequenas partes do código, são executados testes de componente até que eles passem [50].

- **Teste Unitário:** Teste que teoricamente deveria ser o primeiro a ser realizado e analisa cada unidade do sistema (uma unidade é a menor parte do sistema que possa ser testada) separadamente na busca de erros [9]. O Teste de Unidade é a fase de teste que tem como finalidade testar individualmente as funcionalidades do software em questão, garantindo que todas as funcionalidades do software sejam testadas pelo menos uma vez [11].

Sendo assim, os sistemas estão mais complexos e o aumento na exigência de qualidade de software, a resposta rápida e eficaz implicam que os testes de software não possam ser executados manualmente de forma abrangente e segura. Por estes motivos a automatização de testes assumiu uma maior importância relativamente aos testes de software [81].

2.4.1 Teste de Software em Ambientes Ágeis

Testes Ágeis são atividades da etapa de testes num processo de desenvolvimento de software que adotam conceitos de metodologias ágeis. Assim como na atividade de teste de software tradicional, o objetivo principal do teste ágil é garantir a qualidade do software, garantindo o correto funcionamento do sistema, de acordo com as solicitações do cliente. Além disso, Crispin et al. [18] defendem que testes ágeis não significam apenas testes em projetos ágeis, mas testar uma aplicação com um plano para aprender sobre ela e deixar as informações dos clientes guiar os testes; para isso, elas destacam alguns fatores para uma abordagens de teste bem sucedido e ágil:

- Olhar o processo no seu alto nível;
- Colaborar com o cliente;
- Fundamentar as práticas ágeis com o time de projeto;
- Fornecer e obter feedback contínuo;
- Automatizar testes de regressão;
- Adotar uma mentalidade ágil;
- Usar a abordagem considerando que todos são um único time com um único objetivo.

Nos testes ágeis, a equipe não está envolvida apenas em identificar as falhas, mas também em preveni-las. Assim, teste ágil é um desafio para testadores acostumados à estratégia tradicional, principalmente porque eles não precisam esperar pela entrega do sistema para iniciar suas tarefas, mas sim precisam ser proativos e iniciar os testes desde o início do projeto junto com os desenvolvedores [46]. Silva e Moreno [23] estimulam que existe uma interação maior dos testadores com os desenvolvedores que resultará em agir colaborativamente para alcançar os critérios de qualidade esperados pelo cliente.

Segundo Habib [35], é comum empresas que utilizam processos ágeis apresentarem relatos demonstrando quão boas são essas metodologias. No entanto, a maioria dos relatos sobre a utilização das metodologias ágeis não entram em detalhes sobre como funcionam os testes nesses processos. Apesar disso, qualquer um que esteja envolvido em projetos ágeis percebe

que as abordagens tradicionais de teste não funcionam bem nesses casos. Uma vez que, são realizadas entregas com um conjunto pequeno de funcionalidades, as mudanças podem ocorrer com uma alta frequência e assim as atividades de teste precisam se adequar a esse cenário de desenvolvimento. Com a utilização cada vez maior dos processos ágeis, é importante que os profissionais de teste sejam mais qualificados e saibam como trabalhar nesses processos, logo aumenta a necessidade de treinamento da equipe para que todos estejam mais qualificados.

Como nos processos ágeis a cada iteração é entregue uma versão funcional do software, as constantes modificações no sistema podem gerar efeitos colaterais em uma UI (*User Interface*) já testada. Se essa interface teve seus testes automatizados, os efeitos colaterais tendem a ser descobertos com mais facilidade. Assim, toda vez que um teste falha, o testador pode verificar o motivo da falha e reportar o defeito ou, caso seja uma alteração de requisitos, corrigir o teste. Contudo, para que não seja gasto muito tempo com manutenção dos scripts de teste, já que o software estará sob constante alteração, é necessário escolher bem o que automatizar. Nesse contexto, sugere-se que sejam automatizados os testes das UI mais estáveis, pois automações realizadas em interfaces estáveis não sofrem muitos problemas de manutenção.

Segundo Bach [5], automação de teste é muito mais do que fazer com que o computador execute os testes. Existem mais atividades que podem ser automatizadas para facilitar o trabalho da equipe como, por exemplo, Geração de dados e scripts de teste, configuração de sistema, simuladores, gravação de atividades, análise de cobertura de testes, Gerenciamento de Testes etc. Automação está no coração do desenvolvimento ágil de software e é a chave para testes ágeis [60].

2.5 Automação de Teste de Software

Os Testes Automatizados de sistema são desenvolvidos como scripts, com objetivo de garantir a qualidade do sistema, testando as funcionalidades e verificando se estão de acordo com as especificações dos requisitos do sistema e os objetivos esperados [8]. Sendo assim, não existe a dependência da intervenção humana para a execução e se sobressai os benefícios de um computador, como a velocidade de execução, reprodutibilidade exata de um conjunto de

ações, possibilidade de execução paralela de testes, flexibilidade na quantidade, o momento das execuções dos testes e a facilidade da criação de casos complexos de testes. Shamar [61] apresenta que um dos principais motivos que levaram os testes automáticos a assumir uma maior importância, está relacionado com o aumento da complexidade dos softwares e com isso a necessidade de reexecução torna-se mais frequentes. Um exemplo concreto de um script automatizado de teste de sistema para validar a realização do login de uma aplicação web, é apresentando a seguir:

```
1  public class TesteLogin {
2    @Test
3    public void testeLoginGrupo() {
4      WebDriver driver = new FirefoxDriver();
5      driver.get("http://exemplo.com.login");
6
7      driver = driver.findElement(By.id("usuario")).sendKeys("Joao)";
8      driver.findElement(By.id("senha")).sendKeys("123456)";
9      driver.findElement(By.id("login")).click();
10
11     assertEquals("Admin", driver.findElement(By.id("grupos")).
           getText());
12     driver.quit();
13     }
14 }
```

O aumento de capacidade de reutilização deve-se ao fato de ser possível executar os mesmos testes várias vezes, sem contar que pretende-se reduzir o custo de esforço de testar manualmente. Em projetos que existem mudanças/alterações constantes de requisitos, certamente poderá existir a necessidade de executar os mesmos testes mais que uma vez e os testes automáticos são uma boa opção para alcançar esse efeito. A coerência pode ser obtida com os testes automáticos, pois estes podem ser configurados para inserir valores com verdadeiro significado para o negócio ou ambiente em que a aplicação a ser testada se encaixa como, por exemplo, construção de apólice de seguro com valores próximos da realidade [15].

Testes melhores e mais elaborados contribuem para um aumento na qualidade do produto

final. Porém, construir uma suite de scripts de teste automáticos requer uma padronização de atividades e conhecimento em codificação e análise por parte do testador. Para fazer testes automáticos eficientes, caso o teste seja de unidade, é necessário entender a arquitetura do software e a base do código. Caso o teste seja de sistema, é necessário conhecer as regras de negócio, as funcionalidades, e os valores e situações permitidas como resultado.

2.5.1 Limitações da Automação de Teste de Sistema

A utilização da automação de teste também possui limitações, ou seja, podem existir contextos de projetos/empresas que o teste automatizado não seja a melhor solução, por demandar limitações técnicas ou até mesmo ausência de tempo e profissionais especializados em automação. Além disso, um dos desafios na automação de testes é preservar a perspectiva de testar enquanto se desenvolve o teste automático, uma vez que é importante manter a independência entre testar versus desenvolver [2].

Antes de iniciar um processo de automatização de testes de sistema, é importante validar os cenários que serão necessários automatizar, pois caso este levantamento não seja bem feito, o risco de existir scripts automatizados de testes de sistema para validar mais de uma vez o mesmo cenário aumenta. Outro aspecto importante é perceber se o produto em questão necessita de testes automáticos de sistema, caso o produto altere a sua interface com muita frequência, a automatização poderá não ser o processo mais adequado, devido ao fato que os testes teriam de ser refeitos [81]. Moura [15] destaca algumas limitações, dentre elas:

- **Existência de cenários de desenvolvimento que os testes automáticos não substituem testes manuais:** Não é possível testar um software por completo. E neste cenário, o mesmo se aplica para os testes automáticos, uma vez que não é possível automatizar todos os casos de teste por vários fatores, por exemplo, a integração de um sistema com um aparelho externo da aplicação e uso de emuladores. Segundo Fewster [33], alguns dos testes que não devem ser automatizados são:
  - Testes que apenas são executados uma única vez ou raramente;
  - Quando o software é muito volátil, isto é, as funcionalidades mudam dependendo da versão. Com isso o custo de alteração dos testes automáticos é bastante grande;

  - Testes com grau de complexidade muito grande ou impossíveis de automatizar;
  - Testes que envolvem uma interação física como: passar um cartão no leitor de cartões, desconectar/conectar algum equipamento, etc.

- **Existência de cenários de desenvolvimento que os testes automáticos tende a encontrar menos erros:** Se um caso é automatizado, primeiramente esse teste cria-se o caso de teste manual, de modo a verificar que não existem erros. Portanto, se por acaso forem encontrados falhas ou erros, eles serão encontrados primeiro na execução manual do teste. Por definição, todos os testes automáticos tiveram que antes ser executados pelo menos uma vez manualmente [33].

- **Testes automáticos tendem a possuir maior dependência:** É possível que o resultado da execução automática afirme que não detectou erros no teste, uma vez que apenas comparou o resultado obtido com o resultado esperado previamente definido. Se o resultado esperado estiver definido erradamente, ou seja, o script de teste estiver desatualizado com que os requisitos definidos pelo cliente, o resultado final do teste automático não é seguro e a qualidade do teste é questionada, ou seja, os scripts de testes automáticos de sistema possuem uma maior dependência de manter-se atualizado, uma vez que, seguem restritamente o que foi definido no código. Logo, os scripts de testes de sistema automáticos podem, e devem ser revistos e inspecionados para assegurar a sua qualidade.

- **Automação de testes não aumenta a eficácia:** A automação de testes geralmente também custa mais para desenvolver e manter, assim destaca a baixa eficácia [33]. Porém, eventualmente, aumenta a eficiência no sentido de quanto tempo eles demoram a ser executados, uma vez que, depois de implementado, um teste automático geralmente é mais econômico no sentido em que o custo de execução é menor do que um teste executado manualmente.

- **Formas de utilização das ferramentas de automação:** As ferramentas ajudam na estruturação e formatação dos scripts de testes. Porém, os testes automáticos apenas seguirão as instruções de cada passo descrito no script automatizado de teste de sistema, ou seja, se o testador não desenvolver de acordo com o esperado, o resultado do

teste pode ser impactado.

Com a popularização das metodologias ágeis, muitos são os processos de desenvolvimento de software que surgiram trazer mudanças ao modelo tradicional de desenvolvimento, e a atividade de teste de software passou por essa mudança. O processo de teste deve então adaptar-se à agilidade na absorção e implementação antecipada de requisitos. Assim, existe a necessidade da adaptação às práticas ágeis, uma vez que, os testes se constituem de mecanismos para reduzir a ocorrência de problemas e garantir a qualidade do software produzido, consequentemente, assegura a satisfação do cliente. Com isso, a mudança e manutenção dos casos de testes e scripts de testes devem ser realizadas frequentemente.

Portanto, com o alto custo da manutenção dos casos de testes automatizados, a existência desses testes em busca do menor tempo para execução e a demanda por qualidade do sistema passa a existir a necessidade da adequação do melhor momento durante o desenvolvimento do sistema que os testes automatizados devem ser criados e utilizados de maneira produtiva, uma vez que, em ambientes ágeis a mudança dos requisitos é inevitável. Logo, para usufruir das vantagens oferecidas pelos testes automatizados, os engenheiros de teste devem se preocupar-se em encontrar o melhor momento para automatizar os casos de testes.

2.6 Considerações Finais do Capítulo

O capítulo que aqui se encerra, apresentou um breve resumo dos principais conceitos abordados neste trabalho. Os conceitos apresentados se subdividiram entre: conceitos relacionados a pesquisa qualitativa; conceitos relacionados a metodologia da teoria fundamentada, assim como, a confiabilidade entre os codificadores; conceitos relacionados a utilização de métodos ágeis na engenharia de software; conceitos relacionados a utilização de testes de software, assim como sua aplicabilidade utilizando métodos ágeis; e para concluir, conceitos relacionados a automação de teste, apresentando suas vantagens e limitações.

Capítulo 3

Metodologia do Estudo

O objetivo deste capítulo é apresentar os objetivos do estudo empírico realizado para coletar e analisar dados acerca das práticas utilizadas em equipes ágeis para a automação de teste de sistema. Serão apresentadas, além dos objetivos do estudo, as questões de pesquisa e métricas utilizadas. Na sequência, será apresentado o contexto do estudo com detalhamento do universo da coleta e seleção dos participantes. No procedimento experimental, será apresentado como realizamos a coleta dos dados. Por fim, serão apresentadas as etapas da análise dos resultados.

3.1 Perguntas de Pesquisa

Nosso estudo é motivado pelo déficit na literatura quando relacionado ao melhor momento para iniciar a automação de testes de sistema em ambientes ágeis. Consideramos que, ao determinar o melhor momento, as equipes poderão usufruir de amplas vantagens da automação. Assim, foram determinadas as seguintes perguntas de pesquisa.

Pergunta Principal

Qual o momento mais adequado para iniciar a criação dos scripts automáticos dos testes de sistema?

Perguntas Secundárias

- Quais são as dificuldades para automação de testes de sistema?
- Como é realizado o gerenciamento dos artefatos relacionados aos testes de sistema?
- Quais são as dificuldades em manter os testes de sistema?
- Como as mudanças afetam os scripts automatizados de teste de sistema?
- Como são executados os testes de sistema e quais outros testes são executados?

Na visão geral, a metodologia de pesquisa adotada nesse estudo é de origem qualitativa, utilizando a Teoria Fundamentada - TF (em inglês, *Grounded Theory*) [30] para a coleta e análise dos dados. Diante disso, consideramos que o objetivo da pesquisa possui uma relação dinâmica entre o mundo real e o sujeito, ou seja, uma ligação dependente que não pode ser traduzida em números.

Como forma de alcançar os objetivos desta dissertação, selecionamos a opção de entrevista semiestruturada ou livre, no qual o planejamento de perguntas básicas e contextuais do problema abordado faz parte do roteiro para o pesquisador se organizar no processo de interação com o informante. No entanto, após a fase de entrevista ser finalizada, ou seja, existir uma saturação nas respostas, iniciamos uma fase de coleta de dados através do questionário, no qual o mesmo foi construído a partir da análise das entrevistas e serviu para obter uma maior confiabilidade dos dados.

3.2 Contexto do Estudo

A coleta de dados do nosso estudo foi iniciada a partir da realização de convites através de contatos físicos e online para profissionais da área de tecnologia e que estavam integrados em ambiente de desenvolvimento de projeto real utilizando metodologias ágeis, ou seja, projetos de softwares desenvolvidos para empresas em busca de soluções para um problema prático/real. Para isso, entramos em contato através de e-mail, redes sociais (Facebook, Skype e WhatsApp) e até mesmo com pessoas já conhecidas que trabalham atualmente na área de qualidade de software com um objetivo de expandir a rede de contatos.

Após o levantamento da amostra, enviamos através de e-mail, um convite formal, no qual foram apresentadas as informações institucionais, por exemplo: instituição de ensino, programa de pós-graduação, orientadores e orientando, assim como, de forma resumida o objetivo principal da pesquisa e uma pergunta para que o participante selecionasse a opção áudio ou e-mail, como possíveis formas da realização da entrevista. Nesse momento, foram contactados 40 pessoas para participar da pesquisa, entre gerente de projetos e líderes de qualidade que trabalham em grandes empresas nacionais e internacionais, que foram contactados através de listas de discussões sobre qualidade de software e amigos que atuam na área de qualidade de software. Porém, 25 participantes confirmaram a participação, respondendo o e-mail do convite. Durante as entrevistas buscamos informações gerais do processo de desenvolvimento do software e informação específica da atividade de teste de software.

3.3 Procedimento Experimental

No momento em que os prováveis participantes aceitaram o convite de participação, foi enviado um e-mail com o termo de confirmação da pesquisa (Apêndice A). Esse termo serviu para formalizar a aceitação da pesquisa, a partir de informações mais detalhadas de como a entrevista iria ser realizada, e ele possui três seções: (i) informações sobre a pesquisa (dados da pesquisadora, informações dos orientadores e título da pesquisa); (ii) objetivo da pesquisa, com informações mais detalhadas dos problemas identificados, a maneira como a entrevista iria ser realizada, por exemplo: para aqueles que escolheram a forma de entrevista por áudio, o termo de consentimento informava que a mesma seria gravada e posteriormente teria sua transcrição realizada; e (iii) o termo de consentimento, no qual os prováveis participantes precisavam responder o e-mail copiando um trecho de frase, no qual informa que está ciente do objetivo da pesquisa, seu método de entrevista e garantia do sigilo de informações pessoais e profissionais envolvidos. Neste momento, as 25 pessoas que aceitaram participar, responderam o termo de consentimento, logo, formalizaram sua participação na pesquisa.

As entrevistas através da gravação de áudio foram realizadas pessoalmente, via teleconferência ou chamada telefônica. O contato com os participantes foi realizado a partir de uma hora marcada com o mesmo e se dava início a partir de um conjunto base de perguntas (Apêndice B). Contudo no decorrer das respostas surgiam novos questionamentos que foram

sendo apresentado ao entrevistado. Para as entrevistas via e-mail, inicialmente foi enviado um conjunto de perguntas (Apêndice C) e após a resposta do participante, e de acordo com essas respostas foram enviadas novas perguntas, assim, esse ciclo foi concluído quando não havia dúvidas para o pesquisador.

Para as entrevistas realizadas por gravação de áudio, foi necessário realizar a transcrição, ou seja, todo o áudio foi escrito seguindo as regras de transcrição de entrevista [49]. Logo, após o término da gravação, iniciamos o processo de transcrição. Para as entrevistas realizadas via e-mail, foi necessário criar um documento, que contém o agrupamento das respostas (Apêndice D), um exemplo desse agrupamento é: "Na sua experiência, qual momento para automatizar os testes de sistema? R: O mais cedo possível. No caso do nosso processo, os scripts de teste automatizados já podem ser criados antes mesmo do sistema estar pronto, devido aos padrões de codificação utilizados. Quando o sistema (ou parte dele) está pronto para ser testado, então os ajustes necessários são feitos e os casos de teste executados de forma automática.". Portanto, cada entrevista possui um documento com a transcrição e/ou agrupamento, e através dele, realizamos a análise das informações.

Para o experimento, 25 participantes confirmaram a participação através da resposta do termo de consentimento. Dentre estes, 52% optaram por realizar entrevista presencialmente, com áudio gravado, enquanto que 48% optaram por entrevista via e-mail. Dentre os que optaram por e-mail, 33.3% não responderam o e-mail com as perguntas da entrevistas. Portanto, no total tivemos 21 participantes, sendo 13 por entrevista presencial e 8 por entrevista via e-mail, com classificação dos entrevistados como Analista de Teste (8), Líder de Teste (6) Analista de Teste e desenvolvedor (3), Gerente de Projeto (1), Engenheiro de Software (2) e Desenvolvedor (1). Desses, 19 profissionais atuam em equipes (serviço público, empresas e laboratórios de tecnologia em instituição pública de ensino) no Brasil e 2 profissionais em empresas localizada com sede nos EUA.

3.4 Análise dos resultados

Com os documentos de cada entrevista criados, aplicamos a teoria fundamentada [30], no qual foi realizado a seleção dos fragmentos de cada entrevista de acordo com as perguntas de pesquisa, consequentemente a categorização desses fragmentos e definição das áreas te-

mática. Além disso, realizamos o cálculo da confiabilidade partindo de cada pergunta de pesquisa (Figura 3.1). Nas seções abaixo explicamos cada uma das etapas do processo de análise dos dados.

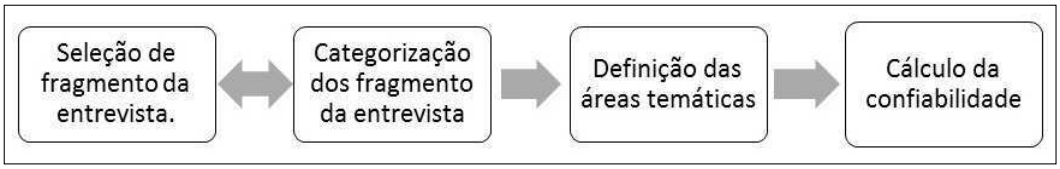

Figura 3.1: Processo da Análise dos Dados

3.4.1 Seleção de fragmentos das entrevistas

A análise foi realizada com foco em responder as questões com a análise final dos dados. Em seguida, realizamos uma leitura geral do documento da transcrição, selecionando fragmento da entrevista que concluímos serem importantes na contribuição do objetivo da pesquisa, assim como, para responder as questões de pesquisa. Essa seleção foi realizada a partir do conhecimento no tema investigado e estava em constante revisão, além disso, realizamos correções gramaticais nos fragmentos, uma vez que são partes da fala do entrevistado. Exemplos de fragmento são apresentados na Tabela 3.1.

Tabela 3.1: Exemplo de fragmento da entrevista

"O maior desafio é você conseguir manter a qualidade do sistema, quando existe muitas mudanças. Geralmente quando o processo é ágil, os Sprints são curtos e o cliente solicita mudança."
"O caso de teste que executamos e que está sempre acontecendo o mesmo resultado, procuramos uma solução para automatizar"

3.4.2 Categorização dos fragmentos da entrevista

Seguimos a proposta da codificação aberta da Teoria Fundamentada [30], na qual os dados são analisados manualmente, submetendo-as ao processo de codificação linha a linha dos fragmentos da entrevista. Para esses fragmentos das entrevistas que expressam a essência do

discurso dos participantes, atribuiu-se categoria, que são palavras ou expressões, formando os códigos preliminares. As categorias representam de maneira simplificada o fragmento da entrevista, que no contexto desta pesquisa, simula uma prática adotada no ambiente de trabalho do participante (Tabela 3.2).

Tabela 3.2: Exemplo da relação Categoria x Fragmento

CATEGORIA	FRAGMENTO
Desafio da qualidade	*"O maior desafio é você conseguir manter a qualidade do sistema, quando existe muitas mudanças. Geralmente quando o processo é ágil, os Sprints são curtos e o cliente solicita mudança."*
Início da automação	*"O caso de teste que executamos e que está sempre acontecendo o mesmo resultado, procuramos uma solução para automatizar."*

Para organizar as categorias com seus respectivos fragmentos, formulamos uma nomenclatura para os fragmentos, seguindo o formato *[ET_NÚMERO DA ENTREVISTA_TIPO DA ENTREVISTA_REFERÊNCIA AO FRAGMENTO]*. Um exemplo do fragmento com nomenclatura é: "[ET_17_LI_37] As mudanças/alterações nos testes automatizados já criados são refeitos.". Os detalhes da nomenclatura são apresentados abaixo:

- **ET** é a numeração que referência a entrevista;
- **NÚMERO DA ENTREVISTA** é a numeração sequencial adotada durante a realização de cada entrevista;
- **TIPO DA ENTREVISTA** é a referência a classificar o tipo de entrevista, "TI"para as entrevistas gravadas via áudio e "LI"para as entrevistas realizadas via e-mail;
- **REFERÊNCIA AO FRAGMENTO** é a numeração que retrata exato tempo ou numeração da linha, no qual o fragmento se inicia.

3.4.3 Agrupamento das categorias em áreas temáticas

Nessa atividade da análise, seguimos a proposta da codificação Axial [54] para aprimorar e diferenciar as categorias resultantes da codificação aberta [27]. Selecionamos as categorias mais relevantes e as inserimos como fenômeno central para estabelecer relações entre as categorias e subcategorias. Os dados, portanto, foram agrupados através das conexões entre as categorias, que representam de maneira simplificada o fragmento da entrevista. Logo, de um conjunto de categorias forma-se as área temática, que abrangem práticas comuns a um contexto, que consequentemente foram extraídas das perguntas de pesquisas (Tabela 3.3).

Tabela 3.3: Exemplo da relação área temática x categoria

ÁREA TEMÁTICA	CATEGORIA
Automação	Início da Automação
	Automação não inicia
	Prática após automação
	Desafios da automação
	Módulos para automação

Após a conclusão do relacionamento da categoria com a área temática de cada entrevista, foi realizada uma revisão desse relacionamento, na qual um 2º pesquisador executou esta atividade, com proposito em validar toda análise realizada. Logo, um convite foi enviado para um voluntário que também pesquisa na área de qualidade de software, ou seja, conhece a problemática investigada nesta pesquisa. Após o aceite, do convite, e de acordo com sua disponibilidade, foi apresentado uma visão geral dos objetivos e metodologia adotada neste trabalho, para que o revisor compreendesse as atividades realizadas e atingisse o objetivo da revisão.

3.4.4 Cálculo da confiabilidade

Após a análise qualitativa das áreas temáticas e categorias resultantes dos dois pesquisadores serem confrontadas, também optamos por analisar a confiabilidade dos dados. Para isso, utilizamos a calculadora Recal3 [28] com base nas categorias e áreas temáticas previamente classificadas.

Para a utilização da Calculadora Recal3 foi necessário seguir critérios como: a utilização de dados nominais; cada arquivo deve representar vários codificadores trabalhando em uma única variável, uma vez que o ReCal3 só pode analisar uma variável de cada vez; cada coluna deve representar o trabalho de um único codificador em uma variável; cada linha deve representar uma única unidade de análise; e todos os códigos devem ser representados numericamente. Sendo assim, para realizar esse cálculo foi criado um arquivo, para cada pergunta de pesquisa, no formato .csv com as categorias relacionadas às áreas temáticas de cada pergunta (eixo x) e cada entrevista (eixo y). Para cada entrevista analisada, foi verificado a presença de cada categoria, assumindo o valor 1 quando existente e o valor 0 quando não. Após a conclusão da criação dos arquivos, compilamos cada um deles na calculadora Recal3, como uma maneira de verificar a conformidade dessa análise e obter a porcentagem média de confiabilidade para cada pergunta de pesquisa de acordo com os fragmentos. Também foi verificado a quantidade de entrevistados que obteve fragmento(s) de sua entrevista relacionado a categoria(s) após a análise da entrevista.

Seguimos a utilização da calculadora Recal3, realizando para cada pergunta de pesquisa o agrupamento das categorias que auxiliaram na formalização da análise das respostas. Para isso, selecionamos dois dos coeficientes de confiabilidade mais populares para dados nominais: Porcentagem de confiabilidade e Kappa de Cohen. A análise qualitativa alcançou um grau substancial de 55% de confiabilidade da análise dos dados, enquanto o valor médio de Kappa de Cohen é 0.52, ou seja, o acordo percentual é considerado aceitável.

3.4.5 Pretensão de resposta as perguntas de pesquisa

A redação da teoria emergente é elaborada por meio de um processo que "consiste numa narrativa descritiva sobre o fenômeno pesquisado" [54]. Essa atividade requereu que possuíssemos habilidades para selecionar as informações importantes para o desenvolvimento do modelo teórico que representa a problemática estudada.

Para que esta teoria tivesse validade, foi necessário comparar os conceitos teóricos estudados e as suas relações com os dados coletados [6]. Contudo, é importante enfatizar que a teoria emergente possui similaridades com as teorias que foram investigadas anteriormente [72] [29]. A obtenção das respostas para as questões de investigação após a análise final dos dados possibilita a criação de novas hipóteses para serem investigadas em trabalhos futuros.

3.5 Considerações Finais do Capítulo

Este capítulo apresentou um conjunto de atividades e técnicas para a formulação do estudo, assim como tornar mais claro todos os passos utilizados até chegar no resultado final dessa pesquisa. Para isso, foram apresentados: as perguntas de pesquisa; a maneira como foi realizada a coleta de dados; a maneira como as entrevistas foram conduzidas de acordo com a amostra obtida; e por fim, todo o detalhamento de cada etapa da fase de análise dos dados.

Capítulo 4

Resultados do Estudo

O objetivo deste capítulo é apresentar os resultados da análise após a etapa de agrupamento das categorias em áreas temáticas. Sendo assim, obtivemos 282 fragmentos selecionados, distribuídos em 24 categorias que foram associadas a 6 áreas temáticas (Comunicação, Documentação, Pré Requisito, Execução de Testes, Automação de Testes e Manutenção de Testes). Cada área temática possui uma quantidade de categorias associadas como: Comunicação com 3 categorias, Documentação com 3 categorias, Pré Requisito com 4 categorias, Execução de Testes com 4 categorias, Automação de Testes com 5 categorias e Manutenção de Testes com 5 categorias (Tabela 4.1). Nas seções abaixo, será apresentada a ideia central de cada categoria de acordo com sua área temática, assim como exemplos de fragmentos que contextualize a análise apresentada.

O relacionamento das áreas temáticas para responder a pergunta de pesquisa principal, assim como as perguntas de pesquisa secundárias, foi realizado de acordo com a ideia central de cada pergunta e dos fragmentos relacionados a cada categoria. Portanto, agrupou as seguintes áreas temáticas para responder cada pergunta de pesquisa secundária: Comunicação e Documentação para a primeira pergunta[1]; Documentação e Pré Requisito para a segunda pergunta[2]; Manutenção de testes para a terceira pergunta[3]; Automação de testes e Manutenção de testes para a quarta pergunta[4]; e Execução de testes para a quinta pergunta[5]. Logo,

[1]Quais são as dificuldades para automação de teste de sistema?

[2]Como é realizado o gerenciamento dos artefatos relacionado a testes de sistema?

[3]Quais as dificuldades de manter os testes de sistema?

[4]Como as mudanças afetam os scripts automatizados de teste de sistema?

[5]Como é realizado a execução dos testes e quais outros testes são executados?

Tabela 4.1: Fragmentos por Área Temática - Categoria

Áreas Temáticas	**Categorias**	**Qtde. Fragmentos**
Comunicação	Priorização dos requisitos	13
	Níveis de responsabilidade	10
	Envolvimento do cliente	25
Documentação	Entendimento da funcionalidade	15
	Desafios da qualidade	15
	Apresentação dos requisitos	14
Pré Requisito	Critérios de Cobertura	3
	Criação dos testes manuais	14
	Papel do teste manual	5
	Análise preliminar para criação de testes	9
Execução de testes	Tipos de teste	8
	Frequência de execução	9
	Testes exploratórios	17
	Utilização de integração contínua	6
Automação de testes	Início da Automação	25
	Automação não inicia	2
	Práticas após automação	4
	Desafios da automação	26
	Módulos para automação	10
Manutenção de testes	Práticas após mudanças	5
	Responsáveis pela manutenção	3
	Efeitos da manutenção	18
	Priorização das alterações	14
	Frequência de mudanças	12

a Tabela 4.2 apresenta as categorias, número de fragmentos, o número de entrevistados, o total de fragmentos e o total de entrevistados relacionado a cada pergunta de pesquisa. Além disso, também apresenta a porcentagem média de confiabilidade das categorias de cada pergunta de pesquisa.

4.1 Comunicação

Esta área temática tem o objetivo de relacionar todas as categorias que contemplem informações que retratem como é o processo de comunicação no processo de desenvolvimento das equipes. Portanto, agrupamos as categorias de Priorização de requisitos, Níveis de responsabilidade e Envolvimento do cliente.

Na priorização de requisitos, observamos que na maioria das equipes em que os entrevistados participam, o cliente não documenta os requisitos no momento de repassar para a equipe, com isso, não existe uma formalidade no momento que os requisitos são apresentados a equipe. Dessa forma, a gerência da equipe possui a responsabilidade de manter o controle do status de cada funcionalidade, como é o cenário do fragmento *"[ET_01_TI_0502] Chega às demandas do cliente, tanto por e-mail, de reuniões ou visitas periódicas, passa por um Product Owner (PO), que filtra aqueles requisitos, faz a prioridade e segue para o Scrum Master onde ele analisa."*. Por outro lado, existem ambientes de desenvolvimento em que a responsabilidade de priorizar os requisitos é realizada com a participação de toda a equipe, de forma que se mantenha a rastreabilidade do que já foi implementado e o que está pendente na lista de backlog, um exemplo desse cenário é o fragmento *"[ET_20_TI_1754] Hoje é feita toda a análise no começo do Sprint, análise essa que é com a equipe de desenvolvimento e a equipe de teste."*, a mesma entrevista ainda completa a ideia em outro fragmento *"[ET_20_TI_0405] Na hora fazemos uma análise caso seja preciso alguma mudança, nós comunicamos com eles (clientes) para marcar uma reunião."*.

Para os níveis de responsabilidade, concluiu-se que para aquelas equipes que possuem papéis definidos, ou seja, existem atividades que são atribuídas para pessoas específicas realizá-las, adotam abordagens de trabalho distintas. Uma dessas abordagem é o acumulo de função, como descrito no fragmento *"[ET_08_TI_1200] Como não temos, e dificilmente conseguimos, um documento já pronto no início, à equipe de qualidade passa a controlar essa parte*

Tabela 4.2: Análise de confiabilidade

Áreas Temáticas	**Categorias**	**PPS-01**		**PPS-02**		**PPS-03**		**PPS-04**		**PPS-05**	
		FR	**R**	**FR**	**R**	**FR**	**R**	**FR**	**R**	**FR**	**R**
Comunicação	Priorização dos requisitos	14%	27%	–	–	–	–	–	–	–	–
	Níveis de responsabilidade	11%	41%	–	–	–	–	–	–	–	–
	Envolvimento do cliente	27%	86%	–	–	–	–	–	–	–	–
Documentação	Entendimento da funcionalidade	16%	50%	20%	50%	–	–	–	–	–	–
	Desafios da qualidade	16%	55%	20%	55%	–	–	–	–	–	–
	Apresentação dos requisitos	15%	50%	19%	50%	–	–	–	–	–	–
Pré Requisito	Critérios de Cobertura	–	–	4%	14%	–	–	–	–	–	–
	Criação dos testes manuais	–	–	19%	59%	–	–	–	–	–	–
	Papel do teste manual	–	–	7%	18%	–	–	–	–	–	–
	Análise Preliminar para criação de testes	–	–	12%	32%	–	–	–	–	–	–
Execução de testes	Tipos de teste	–	–	–	–	–	–	–	–	20%	27%
	Frequência de execução	–	–	–	–	–	–	–	–	23%	41%
	Testes exploratórios	–	–	–	–	–	–	–	–	43%	50%
	Utilização de integração contínua	–	–	–	–	–	–	–	–	15%	18%
Automação de testes	Início da Automação	–	–	–	–	–	–	21%	82%	–	–
	Automação não inicia	–	–	–	–	–	–	2%	9%	–	–
	Práticas após automação	–	–	–		–	–	3%	18%	–	–
	Desafios da automação	–	–	–	–	–	–	22%	73%	–	–
	Módulos para automação	–	–	–	–	–	–	8%	32%	–	–
Manutenção de testes	Práticas após mudanças	–	–	–	–	10%	23%	4%	23%	–	–
	Responsáveis pela manutenção	–	–	–	–	6%	14%	3%	14%	–	–
	Efeitos da manutenção	–	–	–	–	35%	64%	15%	64%	–	–
	Priorização das alterações	–	–	–	–	27%	55%	12%	55%	–	–
	Frequência de mudanças	–	–	–	–	23%	45%	10%	45%	–	–
Total		92	21	75	21	52	21	119	21	40	21
Porcentagem média de confiabilidade		**57%**		**54%**		**48%**		**53%**		**62%**	

Nota: FR: a razão de fragmentos classificados na categoria, R: a razão de entrevistados que proferiram os fragmentos classificados, (PPS: Pergunta de Pesquisa Secundária) PPS01: Quais as dificuldade em manter os testes?, PPS02: Como é o gerenciamento de novos requisitos/mudanças?, PPS03: Como as mudanças afetam os scripts automatizados de teste de sistema?, PPS04: Como os scripts automatizados de teste de sistema são atualizados?, PPS05: Como é realizado a execução dos testes e quais outros testes são executados?

de requisito. E muitas vezes acaba que ficamos atualizando o documento de um lado e eles (os desenvolvedores) já implementando do outro, vai se alinhando no decorrer do tempo." e a outra é a responsabilidade de garantir a qualidade de produtos diferentes, como descrito no *"[ET_01_TI_0636] Nós desenvolvemos três produtos. Então são três equipes de desenvolvimento diferentes, mas é uma equipe de teste para testar os três produtos."*. Nos cenário de ambiente de desenvolvimento que possuem equipes com papéis não definidos, uma pessoa é atribuída para implementar uma funcionalidade e com isso deve garantir a qualidade ou as tarefas são atribuídas por demanda para quem sentir confiança em realizá-las, como descrito no fragmento *"[ET_02_TI_1050] Hoje, não existe mais a equipe de teste separada. O teste está dentro do desenvolvimento, todo mundo trabalha junto."*.

Quando se trata de envolvimento do cliente no processo de desenvolvimento, percebeu-se que é frequente essa participação na maioria das equipes, como deve ser em métodos ágeis. No entanto, existem variações no modo de participação. Por exemplo, cliente que acessa o código construído pela equipe para analisar, como descrito no fragmento *"[ET_01_TI_0729] Ele dá opinião, o nosso cliente é daquele que abre o código e quer ver."*; o cliente possui o contato direto e constante com a equipe através de conversas via teleconferência, e-mail e telefones, com o esclarecimento rápido das dúvidas, como descrito no fragmento *"[ET_01_TI_0710] O cliente está sempre perto, ele está sempre disponível 24 horas para dúvidas ou qualquer coisa que precisamos mandar e-mail e estamos sempre em contato."*; e clientes que são classificados como participativos, mas a comunicação entre equipe e cliente é limitada a uma reunião semanal para esclarecimento de dúvidas, definição de novos requisitos ou mudanças e acompanhamento das atividades gerenciais. No entanto, também existem clientes que não são envolvidos no processo de desenvolvimento, apenas expõem os problemas e uma possível solução, como descrito no fragmento *"[ET_02_TI_2201] O cliente, ele não se envolve muito no fluxo do desenvolvimento até porque para ele isso é transparente."*.

4.2 Documentação

Esta área temática tem o objetivo de relacionar todas as categorias que agrupem informações da maneira como a documentação é realizada processo de desenvolvimento das equipes. Por-

tanto, agrupamos as categorias de Entendimento da funcionalidade, Desafios da qualidade e Apresentação dos requisitos.

No geral, para facilitar o entendimento das funcionalidades, algumas equipes possuem o PO integrado à equipe de desenvolvimento, tornando o esclarecimento de dúvidas mais rápido, como no fragmento *"[ET_14_TI_1225] Eu tenho o meu PO a minha disposição a todo momento que preciso e que tenho alguma dúvida seja relacionado a um comportamento, a uma regra que seja atendida ele está sempre à disposição."*. Encontramos processos de desenvolvimento que possuem documentação bem detalhada, como o fragmento *"[ET_10_LI_14] No início do desenvolvimento, definimos as funcionalidades gerais e o cliente vai definindo as atividades essenciais."*. No entanto, é comum encontrar clientes que não têm conhecimento detalhado do que deve ser criado para resolver o problema, apenas apresenta o problema e a medida que a equipe desenvolve, o mesmo fornece o feedback e com isso não existe documentação, como o fragmento *"[ET_19_TI_0450] O cliente dá a ideia que ele quer, e damos a ideia do que podemos fazer; e ele diz: isso aqui está bom, isso aqui está ruim, não quero isso, pensem de outra forma. Ele só tem a ideia, mas a parte detalhada quem determina somos nós, o cliente informa apenas a ideia de como quer o seu projeto."*.

Quanto aos desafios na qualidade de um sistema, o relacionado da cobertura dos casos de testes com relação aos requisitos do sistema é destacado, ou seja, verificar se todos os requisitos estão sendo validando com a quantidade de casos de testes criadas, como descrito no fragmento *"[ET_03_TI_0900] Para conseguir abranger uma boa cobertura de teste para o sistema que cubra as funcionalidades mais que também seja em um tempo ágil porque as vezes devido as mudanças e aos desenvolvimentos não dá tempo de ir mais devagar."*. A dificuldade na cobertura dos casos de testes em relação aos requisitos está bem relacionada com a ausência de documentação formal no processo de desenvolvimento. Com isso, este foi outro ponto de destaque na entrevista *"[ET_20_TI_1343] Pouca documentação, pouca definição dos requisitos é bem complicado. As vezes tem algo para testar e você não sabe o que testar o que validar, não conhecimento da aplicação e aplicação de alta complexidade de baixo nível também é um pouco complicado."*. Um outro ponto destacado como desafio da qualidade é a organização da equipe para padronizar o processo de planejamento e escrita dos casos de testes/scripts automatizados, como descrito no fragmento *"[ET_19_TI_1105]*

Para mim, a organização da equipe tem que determinar um padrão para escrever os testes.".

Na apresentação dos requisitos, a ausência de documentação compromete o entendimento das funcionalidades. Com isso, geralmente existe uma alta frequência de solicitações de mudanças, inserção de novos requisitos ou até mesmo cobranças de requisitos que o cliente não detalhou o suficiente para o entendimento da equipe, como o fragmento *"[ET_13_TI_0450] Temos uma pequena ideia do que o cliente quer porque geralmente ele chega e fala: olha estou precisando... esse aqui é o meu problema ele chega com problema ele não chega já com requisitos. Ele diz: tenho esse problema aí elaboramos os requisitos, fazemos uma documentação para resolver o problema do cliente.".* Também existem clientes que, ao solicitar o produto, enviam um documento com os requisitos detalhados ou uma descrição mais formal de todas as regras que o sistema deve possuir para que a equipe crie um documento de requisitos detalhados, como o fragmento *"[ET_12_TI_1423] O documento de requisito é feito caso de uso, com detalhamento de regras e o passo a passo do caso de uso a interação com o sistema e o usuário, então iniciamos um primeiro Sprint de acordo com esse documento.".*

4.3 Pré Requisito

Esta área temática tem o objetivo de classificar todas as categorias que contenham informações de como e quando iniciam os testes de sistema e o planejamento da atividade de teste e requisitos, além da definição da importância dos testes para a garantia da qualidade do sistema. Para isso, agrupamos as categorias de Tratamento de fluxos, Criação dos testes manuais, Papel do teste manual e Análise preliminar para a criação de testes.

Nos critérios de cobertura, o desafio de possuir testes que validem todos os requisitos, ou seja, uma cobertura adequada dos casos de teste é sempre alto por causa da instabilidade das funcionalidades e a falta de tempo para manutenção. Com isso, as equipes se organizam para definir caminhos acessíveis, ou seja, não executam todos os casos de testes que foram criados, apenas selecionam os que a equipe define como importante para a garantia da qualidade sem comprometer a qualidade do sistema, como é o cenário do fragmento *"[ET_03_TI_2007] Não pode passar por todos os casos de teste, temos que ver os caminhos mais acessíveis.".* No entanto, existem equipes que buscam estar intercalando a pessoa

responsável pelos testes para evitar a execução de cenários iguais, sem a possibilidade da construção de novos cenários, por exemplo, o fragmento *"[ET_17_LI_44] Cobrir o máximo possível dos cenários e alternar as pessoas que testam o mesmo aplicativo para maior cobertura."*.

A criação dos testes manuais está presente no início de cada Sprint, a partir do momento que os requisitos, funcionalidades ou problemas são apresentados. Nesse momento de criação, surgem dúvidas do comportamento correto de cada funcionalidade e o PO tem um papel fundamental para o esclarecimento das dúvidas, por exemplo, este relato é encontrado no fragmento *"[ET_15_LI_33] O primeiro Requisito Liberado para a versão do produto, dá início ao ciclo de especificação dos Casos de Teste. A primeira Suite de Casos de Teste Liberada dá início ao ciclo de especificação das features (Cucumber, pelo analista de teste do xx)."*. Porém, em alguns cenários de desenvolvimento, algumas equipes não possuem pessoas com responsabilidade na criação dos casos de teste, uma vez que, o cliente disponibiliza funcionários para realizar os testes manuais. Logo, esse cenário se encaixa mais no contexto de testes de aceitação em cima de uma versão entregue ao cliente, como é o caso do fragmento *"[ET_10_LI_99] No meu contexto, testes manuais é de responsabilidade do cliente. Como falei, eles alocaram um funcionário dele para homologar o sistema. Então, toda vez que enviamos uma versão, a funcionário simula atividades/fluxos comuns ao contexto deles. Nós não criamos testes manuais, só automáticos."*.

O papel do teste manual nos métodos ágeis, geralmente são criados no início do Sprint, assim são utilizados como documentação nas execuções posteriores, para seleção dos casos de testes que serão automatizados e como suporte para as equipes que são responsáveis apenas pela automação dos cenários, por exemplo o fragmento *"[ET_02_TI_4145] Vai servir tanto para ele quando para o pessoal da equipe x que for pegar na sua frente depois para testar. Então, nos primeiros dias do Sprint o cara de teste vai escrever lá os roteiros e validam com o PO aquele cenários que ele tá pensando que fazem sentido, então primeira semana do Sprint na primeira semana é muito mas nos primeiros dias o foco dele é esse, fazer a documentação de teste."*. Contudo, os casos de testes manuais e os automatizados são um espelho do que o cliente definiu como requisito, no entanto, os manuais retratam do funcional até a usabilidade do sistema, como destaca o fragmento *"[ET_03_TI_1843] Atividade de teste ela vai ser os olhos do cliente dentro do desenvolvimento do software*

e dessa forma podemos perceber o que ele realmente quer para preencher os requisitos dele e para garantir a qualidade, as vezes algumas usabilidades que o cliente não solicitou mas verificamos.". Além disso, em muitos ambientes de desenvolvimentos, a descrição dos casos de testes manuais, tornam-se a principal documentação do sistema, como o fragmento *"[ET_04_LI_62] O teste, que também tem função de documentação, devem estar sempre alinhados com os requisitos."*.

Durante uma análise preliminar para criação de testes, uma das atividades é a definição de qual tipo de teste adotar após a chegada do requisito, alguns entrevistados destacaram a ideia de verificar a possibilidade de automação no início do desenvolvimento, para isso foi questionado se existia tempo, o quanto a funcionalidade estava definida, como é o caso do fragmento *"[ET_03_TI_2116] Pegávamos os requisitos e a partir daí via o que era possível automatizar de maneira que ajudasse no tempo de teste."*. Outra forma de análise realizada é o mapeamento dos casos de testes manuais que poderiam ser automatizados, como no fragmento *"[ET_03_TI_3616] Você só vai levantar os possíveis casos, no caso títulos dele então você vai pensar neles e juntamente vai pensando com os manuais no início quando chega."*.

4.4 Execução de testes

Esta área temática tem o objetivo de classificar todas as categorias que contenham informações de como a execução dos testes é realizada, dos tipos de testes utilizados e a frequência de execução. Para isso, agrupamos as categorias Tipos de teste, Frequência de execução, Testes exploratórios e Utilização de integração contínua.

Os tipos de teste estão normalmente relacionados com o tipo de ambiente de teste para tentar reproduzir ao máximo o ambiente de produção. Portanto, com a diversidade de ambientes para a execução dos testes, existem as variações dos tipos de testes que são executados (considerando a particularidade de cada equipe), alguns só executam testes manuais de sistema e por isso conseguem separar pequenas funcionalidades para cada suite de teste, por exemplo, a entrevista *"[ET_13_TI_1340] Geralmente executamos pequenas partes, então fazemos alguns testes manuais de sistema e teste de estresse*". Um outro ponto é que normalmente existem equipes que realizam a execução dos testes com base em uma versão do

software e em um ambiente de execução isolado (ambiente destinado a equipe de qualidade), mas também existem equipes que não adotam essa divisão e realizam os testes no mesmo ambiente de desenvolvimento.

Para existir a execução controlada dos testes, as equipes realizam planejamento para execução dos testes, evitando atraso na entrega das funcionalidades. Para isso, a equipe de desenvolvimento tem o compromisso de entregar pequenas versões do decorrer do Sprint, e assim a equipe de qualidade iniciar as execuções dos testes, sejam eles manuais ou automatizados, como é o cenário do fragmento *"[ET_20_TI_1612] Na criação já inicia no planejamento execução é após o pessoal de desenvolvimento terminar as user stories, então já começamos a execução e o report é no final do Sprint, a execução dos testes automatizados utilizamos a integração continua, então sempre que for criada uma versão nova que tenha testes automatizados do Sprint anterior, esses testes são executados então é algo rotineiro."*. Entretanto também existem equipes que não possuem um planejamento bem definido, ou seja, a execução dos testes não possui um prazo definido para iniciar e concluir e assim o processo de teste é totalmente adaptado no decorrer do Sprint.

Os testes exploratórios são frequentemente utilizados pelas equipes de qualidade, pelo fato de ser uma maneira de execução que o custo de tempo é menor, como por exemplo, o cenário de desenvolvimento da entrevista *"[ET_05_LI_216] Utilizamos testes exploratórios com alta frequência (todo Sprint), sempre que ociosos ou após a execução dos testes mais prioritários, não havendo tempo para a execução de todos os casos de teste com menor prioridade."*. Também possuem uma maneira de identificar cenários de inconsistência inesperada, onde o testador não pensou no cenário manual para especificar, uma forma do testador sair do escopo para cruzar áreas do sistema que não sido pensada durante o planejamento dos casos de testes, como destaca no fragmento *"[ET_08_TI_0234] Os testes exploratórios são sempre a melhor saída que é para você conseguir testar de maneira mais ágil com escopo determinado por você, o tanto que vai estar fazendo parte do escopo e o que você não vai conseguir testar."*.

A utilização da integração contínua é uma prática de desenvolvimento de software no qual os membros de uma equipe integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia. Cada integração é verificada por um *build* automatizado (incluindo testes) para detectar erros de

integração o mais rápido possível [26]. Equipes acham que essa abordagem leva a uma significante redução nos problemas de integração e permite a equipe desenvolva software coeso mais rapidamente, por exemplo o fragmento *"[ET_20_TI_1630] Para a execução dos testes automatizados utilizamos a integração contínua, então sempre que for criado uma versão nova que tenha testes automatizados do Sprint anterior esses testes são executados então é algo rotineiro."*.

4.5 Automação de testes

Esta área temática tem o objetivo de classificar todas as categorias que contenham informações sobre a automação de testes de sistema em ambientes ágeis. Para isso, agrupamos as categorias Início da automação, Automação não inicia, Práticas após automação, Desafios da automação e Módulos para automação.

Para algumas equipes, o início da automação é realizado após os responsáveis pelas atividades de qualidade analisarem critérios de interface gráfica, quantidade de execuções manuais, grau de complexidade e estabilidade das funcionalidades. Por exemplo, o fragmento *"[ET_01_TI_1410] A primeira coisa a interface gráfica já está concluída? Quantas execuções manuais mais ou menos? Já rodou umas duas ou três vezes. É possível automatizar? Então está pronto para automatizar."*. Constatamos também que para algumas equipes, o critério para automatizar os casos de testes é sempre estar com a atividade de automação um Sprint atrás do Sprint atual da equipe, evitando retrabalho, ou seja, a equipe de teste automatiza as funcionalidades que foram desenvolvidas no Sprint anterior. Porém, existem equipes que mesmo cientes do retrabalho, iniciam a automação assim que os requisitos são repassados, ou seja, no momento que a equipe de desenvolvimento implementa a funcionalidade, a equipe de qualidade já inicia a criação dos scripts, como no fragmento *"[ET_09_LI_116] O mais cedo possível. No caso do nosso processo, os scripts de teste automatizados já podem ser criados antes mesmo do sistema estar pronto, devido aos padrões de codificação utilizados. Quando o sistema (ou parte dele) está pronto para ser testado, então os ajustes necessários são feitos e os casos de teste executados de forma automática."*.

As esquipes normalmente não iniciam a automação dos testes de sistema quando os requisitos e a interface gráfica não estão totalmente definidos, uma vez que como os scripts

dos testes automatizados possuem uma dependência com os componentes gráficos para serem implementados, como é o cenário do fragmento *"[ET_01_TI_1512] Só que eu comecei a observar que a interface gráfica mudar bastante então cheguei para equipe disse: não vamos fazer mais automação enquanto não tiver a interface gráfica fechada."*.

Como prática adotada após a automação de um caso de teste, na maioria dos cenários de desenvolvimento apresentados nas entrevistas, a execução manual desse teste não é realizada. Com isso, toda atualização que for necessária sempre será realizada no script automatizado e a especificação do caso de teste manual torna-se obsoleta, como no fragmento *"[ET_20_TI_1300] Hoje deixamos de lado os manuais que foram automatizados e considera só a automação, todo Sprint nós fazemos um planejamento do que podem ser alterado, então se tem um Sprint nova e se for alterar um ponto de um sistema que já foi automatizado, temos que voltar todos os testes automatizados e verificar se deve ser atualizado ou não."*. Entretanto, existem cenários que tanto os testes manuais, quanto os scripts automatizados são atualizados, mas no momento da execução, apenas os automatizados são executados e se o script possuir o status de sucesso, a execução do teste manual é desconsiderada.

Um dos desafios da automação mais citados nas entrevistas foi a falta de tempo para criar e manter os scripts automatizados, ou seja, a ausência de atividade no planejamento, mal planejamento ou o ônus intrínseco da atividade do Sprint diretamente relacionada a automação dos testes. No entanto, existiram entrevistas que destacaram apenas o retrabalho e a dificuldade em manter os scripts automatizados atualizados, como é o caso da entrevista *"[ET_11_TI_2454] Manter atualizado os testes automáticos, os testes automáticos estão sendo executados e os casos de uso mudam, então os casos de teste teriam que está espelhados com os testes automáticos."*. Um outro desafio citado foi a capacitação da equipe para utilizar ferramentas de automação muitas vezes é escassa e com isso dificulta o planejamento das atividades e a falta de confiança em determinar quais funcionalidades valem a pena automatizar e assim obter os benefícios da automação como destacou na entrevista *"[ET_02_TI_5022] O principal desafio é medir o que vale a pena automatizar e o que não vale a pena automatizar porque muitas coisas começam a automatizar e eu não encontro muito sentido."*,

Após a decisão de automatizar os testes, as equipes buscam definir os módulos da automação, ou seja, quais as partes do software que terão seus testes automatizados, sendo assim

algumas equipes definem pela parte mais simples do software, aquelas que são fáceis de serem alteradas, como é o cenário do fragmento *"[ET_01_TI_2445] Na experiência em tentar ver aquela funcionalidade que é a base do sistema que você vai ver que não vai alterar, que não vai ter alteração e que tem uma demanda de teste manual para aquilo dali, faz esse fluxo básico, testa esse negócio, funcionalidade são básicas e fácil de alterar que é o núcleo do sistema."*. No entanto, em contextos de desenvolvimento mais críticos, as equipes tendem a definir escopos pequenos para automação, uma vez que nesses cenários o custo com a automação torna-se ainda maior, um exemplo dessa prática é a entrevista *"[ET_07_LI_37] Embora os testes automatizados são mais difíceis quando o sistema é complexo, envolvendo tópicos comunicar com os outros processos etc, você deve sempre tentar testar pequenas escopos em primeiro lugar."*. Entretanto, pensando no custo com a manutenção na automação dos testes, existem equipes que optam pela automação quando existe a frequência na validação, pensando no custo com o reteste para validar os bugs e na execução da regressão dos testes.

4.6 Manutenção dos Testes

Esta área temática tem o objetivo de classificar todas as categorias que contenham informações sobre as práticas e problemáticas da manutenção dos casos de testes e/ou scripts de testes. Para isso, agrupamos as categorias Práticas após mudanças, Responsáveis pela manutenção, Efeitos da manutenção, Priorização das Alterações e Frequência de mudanças.

Práticas após mudanças, para algumas equipes de desenvolvimento, é a realização do planejamento para definir prioridade, impacto e até mesmo alteração no cronograma da entrega, como é o cenário do fragmento *"[ET_09_LI_134] No nosso processo de desenvolvimento, toda mudança solicitada pelo cliente deve ser acordada e documentada. Se há necessidade de mudança de especificação, então é analisado o impacto dessa mudança e o cronograma é alterado, caso necessário."*. No entanto, existem equipes que não realizam planejamento do impacto das mudanças de requisitos e já iniciam as alterações nos casos de teste e scripts de testes de acordo com a solicitação enviada, como o fragmento *"[ET_17_LI_37] As mudanças/alterações nos testes automatizados já criados são refeitos."*.

Os responsáveis pela manutenção dos testes normalmente é responsabilidade da mesma

equipe que os criou, como no fragmento *"[ET_15_LI_48] Desde que comunicadas, as mudanças são encaradas pela equipe com naturalidade, uma nova tarefa de modificação baseada nos Casos de Teste ora modificados é criada e atribuída ao analista de teste para os reparos."*. Porém, não muito comum, existem equipes que possuem uma equipe específica para manutenção dos testes, como o fragmento *"[ET_02_TI_3417] Esse fluxo que eu estou te falando é o fluxo da manutenção evolutiva ou seja do desenvolvimento de sistemas. Nós temos outras equipes e outros times que trabalham com solução da manutenção corretiva ou seja erros que geram no cliente."*.

Os efeitos da manutenção no processo de desenvolvimento geralmente têm um impacto grande. Logo ,destacamos impacto nos scripts de testes que podem tornar-se desatualizados ou obsoletos, os casos de teste também ficam desatualizados e com isso há uma dependência da experiência do testador na funcionalidade, e como consequência de manter os requisitos e casos de testes atualizados, a automação dos testes não é realizada. Um exemplo desse efeito é o fragmento *"[ET_01_TI_2845] Testes automatizados nível de sistemas que tem uma dependência muito grande da interface de um sistema web principalmente o nível de automação dos Testes e da complexidade das alterações aumenta. Então, fica mais complexo de manter isso."*. No entanto, também percebemos equipes que expõe que a mudança não tem efeito/consequência grande no desenvolvimento das funcionalidades, como o fragmento *"[ET_14_TI_1947] No que eu tenho percebido, as mudanças que acabam ocorrendo são muito mais no sentido de novas features."*.

Depois que a mudança é enviada para equipe, necessita-se realizar a priorização das alterações que serão implementadas com urgência, consequentemente os casos de testes que deverão passar por mudanças. Portanto, a realização de reuniões com participação de toda a equipe para essa definição é um cenário presente no fragmento *"[ET_05_TI_1949] Toda vez que ele faz alguma alteração, nós temos algumas reuniões e conversa um pouco e tenta estruturar um pouco o que ele quer e temos que revisar todos os casos de teste já criados para deixar em conformidade com essa nova especificação dele."*. No entanto, também existem equipes que partem logo para alteração dos casos de testes manuais sem realizar a análise preliminar do efeito das mudanças, deixando a atualização dos scripts de testes para uma realização futura, como o fragmento *[ET_02_TI_6457] Basicamente minha atitude como testador nesse cenário seria focar em teste manuais e posteriormente a isso, uma vez*

que a entrega ter sido feita, mesmo apenas com testes manuais, eu iria atualizar meus scripts, ou seja, pegar aquilo que eu tinha e mudar para essa nova realidade, refatorando o código que tinha automatizado, ou seja, pegar e refatorar. E com aquilo que eu tinha manual lá, eu ajustava meus casos de teste automatizados.

A frequência de mudanças que são solicitadas é um aspecto que depende muito do cliente e tipo de sistema que a equipe tem, ou seja, os clientes não estão certos quanto as funcionalidades requeridas pelo sistema e com isso solicitam mudanças frequentemente, a ponto de durante uma implementação da mudança solicitada, o cliente já solicita outra dentro da mesma funcionalidade, como o fragmento *"[ET_02_TI_3355] As mudanças são constantes por conta dessa realidade que eu te falei que hoje chegamos com muita coisa não definida, como processo redondo não é para ter tanta mudança mas hoje nós temos muitas."*. No entanto, também existem equipes que definem suas as atividades para o Sprint, ao ponto de quando existir solicitação de mudança, a mesma é realocada para o próximo Sprint, como o cenário da entrevista *"[ET_04_LI_102] As vezes acontece, mas geralmente é colocada para o próximo Sprint."*. Mas também encontramos cenário de desenvolvimento que a solicitação de mudanças por parte do cliente é rara, por sempre fechar todas as funcionalidades que serão entregues antes de iniciar o desenvolvimento.

4.7 Considerações Finais do Capítulo

Este capítulo apresentou a contextualização para cada área temática classificada, de acordo com a análise das entrevistas. Preocupamo-nos em mapear as ideias centrais discutidas nas entrevistas e cobertas no contexto de cada área, desta forma foram utilizados fragmentos das entrevistas para exemplificar essas ideias.

Capítulo 5

Análise dos Resultados

Este capítulo discute as principais conclusões do estudo, a partir da codificação das entrevistas realizadas com profissionais de tecnologia através da Teoria Fundamentada. O capítulo se organiza pelas perguntas de pesquisa secundária apresentadas no Capítulo 3, relacionadas a (1) dificuldades na automação de testes, (2) o gerenciamento dos artefatos relacionado à testes de sistema, (3) a dificuldade em manter os scripts de testes de sistema, (4) o efeito das mudanças nos scripts automatizados e (5) a realização da execução dos testes de sistemas, assim como (6) a execução de outros tipos de testes. Também realizamos uma análise final para a pergunta de pesquisa principal que busca responder qual o momento mais adequado para iniciar a criação dos testes automatizados de sistema, além de finalizar com um relato das ameaças à validade.

5.1 PPS_01 - Quais são as dificuldades para automação de testes de sistema?

No geral, 11 entrevistados, abordou na sua entrevista as dificuldades na automação, logo, 92 fragmentos foram selecionados para contextualizar e formalizar a resposta para essa pergunta. Assim, observamos entre as dificuldades mais presentes nas equipes ágeis:

- Medição da cobertura dos scripts de teste de acordo com os requisitos para definir o que está sendo validado com a automação dos testes de sistema;
- Validação dos requisitos, visto que, devido a ausência de documentação, por falta de

formalismo no repasse dos requisitos, os scripts de teste podem possuir confiabilidade baixa;

- A definição para o início da automação dos testes de sistemas, uma vez que a equipe deve se basear nas expectativas do cliente;
- A ausência de profissionais com experiência em automação de teste de sistema nas equipes de qualidade;
- O contato frequente com cliente possui a vantagem da rapidez no esclarecimento de dúvidas, mas muitas vezes essas dúvidas servem para que o cliente identifique cenários, que antes não tenham sido especificados, ou seja, antecipe alterações não previstas pela equipe.

Neste contexto, observamos uma diversidade no ambiente de trabalho em relação à abordagem de assimilação dos requisitos. Sendo assim, o gerenciamento dos requisitos e das mudanças existe de maneira formal apenas nas equipes que adotam processo tradicionais (não ágeis). Essas equipes possuem critérios de priorização de mudanças, assim como um controle de *backlog* para as novas funcionalidades, mesmo que seus clientes não facilitem o repasse dos requisitos funcionais. Entretanto, o que se constatou na prática foi um gerenciamento informal ou *ad hoc*, ou seja, tudo que o cliente solicita é desenvolvido de imediato, mesmo que exista impacto nas demais funcionalidades, além disso, as equipes não possuem a documentação ou a mesma é mal elaborada. Um exemplo desse cenário é o fragmento *"[ET_11_TI_0808] Temos um processo não muito bem definido de teste porque justamente pelo modelo que até o próprio cliente tem de às vezes você pede para fazer uma funcionalidade e já na outra semana já estar mudando e fazer um teste novamente pra isso, as vezes não atualiza os testes, os um teste manual sem seguir nenhuma documentação, só a especificação que o cliente passa para os desenvolvedores."*.

O gerenciamento formal de novos requisitos ou mudanças em geral é inviável, uma vez que a comunicação com o cliente em geral é falha, esporádica e normalmente não acompanha o ritmo de desenvolvimento. Porém, independente do nível de envolvimento do cliente, a diversidade na maneira como os requisitos funcionais são repassados é uma característica em comum no desenvolvimento das equipes. Diante disto, o entendimento das funcionalidades

torna-se comprometido pela ausência de um documento detalhado com todas as regras de negócio e fluxos (exceção e alternativos).

O início da criação dos scripts de teste tem um custo alto, uma vez que é necessário o conhecimento técnico, ou seja, experiência nos *frameworks* de automação e definição da arquitetura dos testes. No entanto, nas equipes de qualidade é comum encontrar profissionais que não possuem experiência em automação de teste e normalmente não gostam de programar, sendo assim demanda um tempo maior que o normal para a realização dessas atividades, logo, é necessário um planejamento mais detalhado para não impactar as demais atividades da equipe de qualidade e decorrer atrasos para entrega das funcionalidades ao cliente o que significa um ganho de tempo nas atividades de automação dos próximos Sprints. Um exemplo é o fragmento *"[ET_12_TI_2500] Então a maior dificuldade seria a capacitação dos profissionais para automatizar os scripts"*.

Diante das dificuldades apresentadas, percebemos que o problema se inicia desde a falta de formalismo do cliente ou PO no repasse dos requisitos para equipe, até à ausência de profissionais capacitados para esta atividade. Sendo assim, existe uma complexidade na busca, de uma solução que atenda todos esses problemas, até porque uma vez que parte desses problemas estão relacionados ao contato com cliente. Logo, é preciso cautela quando se refere ao contanto com o cliente.

5.2 PPS_02 - Como é realizado o gerenciamento dos artefatos relacionados aos testes de sistema?

No geral, 9 entrevistados, abordou na sua entrevista informações sobre o gerenciamento dos artefatos, logo, 75 fragmentos foram selecionados para contextualizar e formalizar a resposta para essa pergunta. Com isso destacamos que, o gerenciamento dos artefatos relacionados aos testes de sistema concentra-se apenas na documentação de teste, uma vez que, em geral, a ausência de documentação funcional é uma característica das equipes que utilizam a metodologia ágil como padrão. Nesse sentido, na maioria das equipes das que as pessoas entrevistadas fazem parte, os casos de testes e/ou scripts de testes automatizados acabam sendo a única documentação funcional. Um exemplo é o cenário do fragmento *"[ET_02_TI_4151] Então nos primeiros dias da Sprint o testador escreve os roteiros e está validando com o*

PO aqueles cenários que ele está pensando que fazem sentido. Então, primeira semana do Sprint o foco é fazer a documentação de teste, porque normalmente torna-se a principal documentação.".

O gerenciamento de artefatos também afeta o critério de cobertura dos testes, uma vez que é difícil mensurar quais os requisitos que estão sendo validados, ou seja, com pouca definição dos requisitos, a garantia da validação tende a ficar mais complicada devido à falta da visão específica do sistema. Sendo assim, geralmente as equipes não controlam a cobertura dos scripts de testes por não possuir um documento base para seguir.

No geral, o gerenciamento dos artefatos se restringe à documentação dos testes. Logo, compara-se o problema com a ausência de formalismo no repasse dos requisitos e com isso, o controle com relação ao efeito das mudanças ou definição de novos requisitos funcionais nos casos de testes manuais e/ou scripts de testes automatizado não existem. Portanto, a ausência de gerenciamento dificulta a definição do início da automação dos testes de sistema.

5.3 PPS_03 - Quais são as dificuldades em manter os testes de sistema?

No contexto geral, 9 entrevistados, abordou na sua entrevista informações sobre as dificuldades em manter os testes de sistema atualizados, logo, 52 fragmentos foram selecionados para contextualizar e formalizar a resposta para essa pergunta. Sendo assim, constatamos que as equipes de qualidade definem como difícil manter os casos de testes e os scripts de testes automatizados de sistema acordo com as regras atuais repassadas através do cliente, uma vez que as mudanças realizadas ao longo do processo de desenvolvimento são inevitáveis, e caso não sejam gerenciadas, podem causar um impacto não previsto no projeto. Diante da complexidade em gerenciar as mudanças solicitadas, geralmente as equipes não realiza um planejamento das atividades de qualidade contemplando o custo de tempo para realizar a manutenção na especificação dos testes e nos scripts automatizados. Portanto, essa complexidade inicia a partir da falta de documentação, seguindo da dificuldade no acompanhamento das diversas solicitações de mudanças vindas do cliente, no qual muitas vezes o tempo é curto entre uma requisição e outra, a ponto da primeira solicitação não ser concluída e levando a comprometer o versionamento do software. Em contrapartida, existem

equipes que possuem os requisitos bem definidos e a frequência de mudança é baixa, com isso, não possuem problemas com a manutenção dos testes como por exemplo o fragmento *"[ET_10_LI_81] As mudanças são raras acontecer. Normalmente fechamos bem a funcionalidade antes de começar a implementação."*.

Uma maneira encontrada por algumas equipes para solucionar ou minimizar o problema da manutenção dos testes, foi em acordo com o cliente ou PO priorizar as mudanças, realizando-as em sequência e em pequenas partes como por exemplo o fragmento *"[ET_01_TI_0502] Chega às demandas do cliente chega tanto por e-mail ou de reuniões ou visitas periódicas, passa por um PO, filtra aqueles requisitos, faz a prioridade vai para o Scrum Master onde ele analisa."*. Entretanto, também existem equipes que não possuem essa opção e aceitam todas as solicitações de mudanças. Com isso, surge a consequência dos scripts automatizados desatualizados ou até mesmo obsoletos, até a equipe de qualidade possuir tempo disponível para atualizá-los. Os efeitos surgem com maior impacto na automação dos testes de sistema, por causa da dependência dos elementos de tela e para iniciar a criação dos scripts é necessário no mínimo existir uma versão estável da aplicação como o fragmento *"[ET_01_TI_1512] Só que eu comecei a observar que a interface do usuário (UI) era alterada bastante. Então falei para a equipe que não iriamos fazer mais automação enquanto não tiver a interface do usuário fechada."*.

Com intuito de diminuir as dificuldades de manutenção dos artefatos, supõe-se como provável solução, a necessidade do amadurecimento dos requisitos antes do início da criação da documentação de teste. Ou seja, a especificação manual assim como os scripts de teste automatizado de sistema apenas deve iniciar sua criação após esse amadurecimento dos requisitos, assim as regras de negócios estarão sendo validadas de maneira correta, conforme o definido e sem a ocorrência de alterações com grande impacto nas demais suites de testes.

5.4 PPS_04 - Como as mudanças afetam os scripts automatizados de teste de sistema?

No geral, 10 entrevistados, abordou na sua entrevista informações sobre como as mudanças afetam os scripts automatizados de teste de sistema, logo, 119 fragmentos foram selecionados para contextualizar e formalizar a resposta para essa pergunta. Sendo assim, constatamos

que a criação dos scripts dos testes automatizados requer experiência em ferramentas e *frameworks* utilizados, ou seja, a necessidade de existir conhecimentos específicos. No entanto, as equipes de qualidade muitas vezes possuem testadores sem experiência em automação de teste ou que não gostam de programar, fazendo com que os scripts de testes sejam criados utilizando práticas que não facilitam sua manutenção, ou seja, sem a utilização de design dos testes. Diante das análises realizadas, concluímos que a ausência de boas práticas na criação dos scripts automatizados torna-se um critério com alto impacto no custo da manutenção desses scripts.

Para tentar amenizar o efeito das mudanças nos scripts automatizados, equipes acabam adotando uma estratégia de modularizar o sistema, ou seja, classificam o sistema em módulos com um conjunto de funcionalidades diretamente dependentes. A partir dos módulos definidos, as equipes classificam individualmente o nível de maturidade das funcionalidades que contemplam cada módulo, assim iniciam o processo de automação dos testes. No entanto, para algumas equipes, a frequência das solicitações de mudanças não influência no processo de qualidade, logo, ao existir uma solicitação de mudança, imediatamente são realizadas as alterações nos scripts de teste. Também identificamos equipes em que as solicitações de mudanças não possuem um grande impacto, uma vez que essas mudanças são apenas ajustes e/ou novas funcionalidades, assim, os scripts de teste são levemente modificados. Um exemplo desse cenário é o fragmento *"[ET_13_TI_0855] Ele solicita pequenas mudanças, pequenos ajustes..."*.

Normalmente é comum encontrar equipes com solicitações de mudanças que afetam o processo de desenvolvimento, a ponto de descartar a utilização da automação de teste por ausência de tempo para manter os scripts atualizados. Sendo assim, para evitar problemas com manutenção, assim como considerar o amadurecimento dos requisitos, a criação dos scripts de teste de sistema deve ter sua atividade iniciada após a modularização do sistema, uma vez que após essa atividade supõem-se a matriz de rastreabilidade das funcionalidades esteja criada, e quando existir solicitação de mudanças, todo o impacto será facilmente identificado e assim será possível realizar um novo planejamento das atividades, ou seja, analisar qual o tempo para realizar as atividades com o impacto das mudanças e verificar se a mudança impactará no prazo para entrega ao cliente.

5.5 PPS_05 - Como são executados os testes de sistema e quais outros testes são executados?

No geral, 7 entrevistados, abordou na sua entrevista informações sobre como são executados os testes de sistema e quais outros testes são executados, logo, 40 fragmentos foram selecionados para contextualizar e formalizar a resposta para essa pergunta. Com isso, identificamos apenas a utilização de teste exploratório como teste complementar nos processos de teste utilizados nas equipes, no qual geralmente é executado em uma versão mais estável do sistema, de maneira manual pelo testador e com base na sua experiência nos requisitos do sistema, ou seja, o testador se coloca na posição de um usuário final e não segue nenhuma documentação como por exemplo, os casos de testes. Sendo assim, normalmente, foi o tipo de teste presente no cenário de desenvolvimento em várias equipes de qualidade. O teste exploratório é um teste para ambientes de desenvolvimento que não possuem um processo de qualidade definido e/ou possuem dificuldades em gerenciar os artefatos por causa das solicitações de mudanças. Nesse sentido, existem equipes que apenas utilizam o teste exploratório para minimamente garantir a qualidade do sistema, entretanto também existem equipes que utilizam testes exploratórios como maneira de complementar o processo de qualidade como por exemplo o fragmento *"[ET_03_LI_289] Em toda execução fazemos, uns testes exploratórios adicionais e caso seja um tempo muito curto para execução, fazemos apenas exploratórios."*.

As equipes que utilizam o teste exploratório como complementação, identificaram que por mais que existam especificação dos testes manuais e scripts de teste automatizado, o teste exploratório normalmente ainda é o que encontra mais defeitos no sistema. Também destacamos a importância do teste exploratório no processo de qualidade do software com a justificativa que existem circunstâncias inesperadas e/ou não planejadas para os casos de testes, e que apenas serão encontradas na execução dos testes exploratórios, levando em consideração a experiência e conhecimento nos requisitos do testador.

O teste exploratório também é utilizado com propósito de validação de elementos da tela (interface do usuário-UI), ou seja, as equipes não planejam cenários na especificação manual para verificar por exemplo, se o botão está da cor e na posição definida na prototipação de tela como no fragmento *"[ET_14_TI_3251] No teste exploratório tratamos quando esta-*

mos tentando identificar por exemplo alguma inconformidade de posicionamento de botão principalmente porque tivemos um upgrade de layout agora..."

5.6 Pergunta Principal: Qual o momento mais adequado para iniciar a criação dos scripts automáticos dos testes de sistema?

Com base nas respostas das perguntas secundárias e no processo completo de coleta e análise dos dados, podemos concluir que o início da criação dos scripts automáticos dos testes de sistema possui uma variação de acordo com a prática ágil adotada no processo de desenvolvimento, o produto desenvolvido, a forma como os requisitos funcionais do sistema é repassada e a formação da equipe. E essa variação é encontrada na Tabela 5.1.

Tabela 5.1: Cenário de desenvolvimento x Abordagem na automação de testes

CONTEXTO	**NÍVEL DE ESPECIFICAÇÃO**	**ABORDAGEM MAIS COMUM PARA INICIAR AUTOMAÇÃO**
Equipe de qualidade integrada	Formal	Definição de estabilidade da funcionalidade e execução dos testes manuais em Sprint(s) anterior(es).
	Informal	Posterior a primeira execução manual dos testes durante um Sprint.
Sem equipe de qualidade	Formal	Indefinido
	Informal	Indefinido

Para equipes de desenvolvimento que possuem uma equipe de qualidade integrada no mesmo ambiente da equipe de desenvolvedores, é mais viável iniciar essa criação após o líder da equipe de qualidade perceber que o requisito está estável (não sofrerá grandes mudanças) e a suite de testes manuais foi executada em Sprints anterior(es), assim o risco desses scripts ficarem desatualizados é baixo, o custo com a execução dos testes também diminui

e a manutenção será pontual, sem muito impacto nas demais funcionalidades. Por exemplo, uma funcionalidade implementada e validada através da execução manual dos testes em um Sprint e a entrega ao cliente, o mesmo não identificou a necessidade de grande alteração; assim essa funcionalidade está apta para iniciar a criação dos scripts automatizados no próximo Sprint como no fragmento *"[ET_20_TI_1930] No final de uma sprint onde está tudo entregue tudo fechado que não vai ter alteração, ai sim você começa a automatizar..."*.

Continuando com a ideia da existência de uma equipe de qualidade integrada a equipe de desenvolvimento, existe o cenário que existe uma definição formal de toda a necessidade do cliente e pessoas especializadas em automação de teste de sistema na equipe de qualidade. Para esse cenário de desenvolvimento, percebemos que a criação dos scripts de testes de sistema posterior a primeira execução manual dos testes durante um Sprint, é válida. Assim, durante o período de regressão durante a Sprint é realizada através da execução automática dos testes de sistema. Por exemplo, supomos que existe uma equipe de qualidade que trabalhe com entrega contínua para validação e seja formada por 4 pessoas, sendo 2 dessas especialista em automação de teste de sistema; então na medida que os casos de testes tenham sua escrita e primeira execução concluída, o(s) especialista(s) iniciam a atividade de criação dos scripts de testes automatizados de sistema.

Porém, para aquelas equipes de desenvolvimento que não possuem equipe de qualidade, ou seja, não tem especialistas de qualidade e todos os membros da equipe adotam papel de desenvolvedor e testador, não conseguimos concluir qual o melhor momento para iniciar a automação de testes de sistema, uma vez que esse cenário não adota versionamento do sistema para validação, os profissionais tem que dividir seu tempo para implementar e validar as funcionalidade e normalmente não possuem conhecimento técnico nos *frameworks* e práticas de automação de testes de sistemas.

5.7 Ameaças

Como o trabalho desenvolvido trata da identificação de fatores para auxiliar a equipe de qualidade em definir o momento mais adequado para iniciar a criação dos scripts de testes automatizados em um ambiente de desenvolvimento que utiliza métodos ágeis, necessita ser melhor investigado com maior variação de cenários de desenvolvimento. Porém, devido a

limitação de tempo, realizamos a análise dos dados utilizando apenas uma maneira de coleta de dados, ou seja, através de entrevistas.

Outra limitação do nosso trabalho trata da quantidade das entrevistas realizadas. Para obter os resultados, necessita-se primordialmente dos relatos práticos de profissionais que participam de equipes de desenvolvimento, no qual através da exposição das problemáticas e práticas adotadas no dia a dia; então, para conseguir a disponibilidade de voluntários para participação da pesquisa é muito custoso. Portanto, o número final de entrevistados foi abaixo do que esperávamos, por se tratar de uma pesquisa qualitativa e requerer uma análise mais imparcial, objetiva e representativa. No entanto, o número é compatível com outros estudos relacionados, como por exemplo [21, 75].

Dado que o número de entrevistas que utilizamos nos estudos é restrito e não ocorreu validação prática dos critérios identificados, não podemos garantir que os resultados deste trabalho sejam aplicáveis para toda e qualquer equipe de qualidade que utilizam métodos ágeis. Procuramos variar os mais diversos ambientes de desenvolvimento, englobando equipes ágeis de outros estados do Brasil e fora do País e de tamanho variado. Porém, dado que o contexto deste trabalho é vasto, equipes de desenvolvimento com contextos diferente dos analisados podem não ter sido analisada. Sendo esta, mais uma limitação do trabalho.

5.8 Considerações Finais do Capítulo

Este capítulo apresentou as respostas das perguntas de pesquisa apresentadas no Capítulo 3 com base no mapeamento realizado através da Teoria fundamentada. Para PP1 concluímos as principais dificuldades para automatizar os testes de sistemas. Na PP2, concluímos que a realização do gerenciamento dos artefatos muitas vezes não existe, além disso geralmente a única documentação produzida é a especificação de teste. Na PP3, concluímos que, para diminuir a dificuldade da manutenção dos testes criados, supõe-se o amadurecimento dos requisitos antes de iniciar a criação da especificação de testes, pois promove a diminuição do impacto nas funcionalidades que foram validadas. Na PP4, concluímos que, como os testadores criam os scripts de testes não seguindo boas práticas e consequentemente os scripts ficam difícil de manter, o que muitas vezes faz com que sua utilização seja postergada para um outro momento do desenvolvimento ou extinta do processo de validação. Na

PP5, concluímos que a execução dos testes exploratórios encontra-se presente em processo de qualidade de muitas equipes, ou seja, o mesmo é utilizado para complementar o processo ou como única execução de teste para a garantia da qualidade do sistema. Por fim, na pergunta principal concluímos que existe uma diversidade na abordagem da criação dos scripts dos testes de sistema de acordo com a prática ágil adotada no processo de desenvolvimento, o produto desenvolvido, a forma como os requisitos funcionais do sistema é repassada e a formação da equipe.

Capítulo 6

Considerações Finais

Neste capítulo, apresentamos as conclusões finais do estudo com a discussão das últimas considerações da análise dos dados, assim como, pressupomos algumas conclusões que requerem uma validação futura. Por fim, definimos prováveis trabalhos futuros.

6.1 Conclusões

O trabalho apresentado neste documento se propõe a identificar práticas que definem o melhor momento para iniciar a automação dos testes de sistema, fazendo uso de princípios da análise qualitativa dos dados, mais especificamente, utilizando a teoria fundamentada e o cálculo de confiabilidade dos coeficientes. Para atingir tal objetivo, realizamos um estudo na literatura em conjunto com as observações do dia a dia em ambiente de desenvolvimento para prover argumentos concretos da problemática estudada. Desse estudo, foi determinado o escopo do problema, o público que iríamos abordar e a metodologia utilizada. A definição desses critérios auxiliou na definição do processo de coleta de dados, através de entrevistas e análise dos dados e das métricas da teoria fundamentada, assim como, a definição da pergunta de pesquisa principal e das perguntas secundárias. Em segundo plano, ainda referente a coleta de dados, iniciamos a definição das perguntas que serviram como base para as entrevistas.

Para avaliação dos elementos produzidos neste trabalho, um conjunto de 21 entrevistas foi realizada com indivíduos de contextos de trabalho distintos, por exemplo, equipes de desenvolvimento que possuíam equipe de qualidade para realização da validação e verifi-

cação das funcionalidades e equipes que o próprio desenvolvedor era responsável por esta atividade.

Com a utilização da teoria fundamentada, realizamos a avaliação das áreas temáticas, categorias e fragmentos de todas as entrevistas. Logo, identificamos no estudo resultados satisfatórios com evidências de práticas na automação de teste em processos que adotam métodos ágeis como, por exemplo: (1) Controle e gerenciamento das alterações solicitadas pelo cliente, ou seja, realizar um replanejamento rapidamente caso a solicitação venha a causar um grande impacto na funcionalidade em fase de validação; (2) Adotar práticas de rastreabilidade em relação as funcionalidades, casos de teste, scripts de teste e bugs, com isso, a alteração é facilmente identificada, assim como as funcionalidades que indiretamente podem ser afetadas com a resolução de um bug; (3) Início da criação dos scripts de teste automáticos após o requisito funcional estabilizado e realização de execuções manuais dos casos de teste, com isso, o custo com a manutenção tende a se tornar menor e o gerenciamento das alterações podem ser facilmente realizados, mantendo os testes sempre de acordo com os requisitos definidos pelo cliente; e (4) Início da atividade de criação dos scripts de teste automáticos de sistema é planejado para ocorrer sempre em um Sprint posterior à Sprint de desenvolvimento, ou seja, os testes manuais devem ser executados no Sprint de desenvolvimento. Nesse momento, bugs mais críticos foram identificados e possivelmente resolvidos, logo a automação tem o objetivo de manter a constante validação da funcionalidade, identificando bugs rapidamente.

Com as informações obtidas nesta dissertação e com base nos relatos apresentados durante as entrevistas, percebemos que normalmente as equipes que dizem utilizar métodos ágeis, ainda adotam algumas práticas da metodologia tradicional. Na verdade existe a cultura que impede inovar com a utilização de novas abordagens ou até mesmo, a equipe prefere ficar na zona de conforto, para não correr riscos. Sendo assim, a atividade de automação de teste de software de sistema, se enquadra nesse contexto, uma vez que geralmente os testadores não gostam de programar e com isso não buscam novos conhecimentos em boas práticas de automação de testes, facilitando e aprimorando o processo de criação e manutenção dos scripts de testes automatizados de sistema. Percebemos também que, como existe problema com o tempo, muitas vezes as equipes postergam o início da criação desses scripts de testes automatizados de sistema a ponto de nunca iniciar esta atividade e com isso prevalece apenas

a criação e execução dos testes de sistema manuais.

Portanto, assim como Alégrotha [1] apresentou em seu trabalho, o custo com a manutenção e o custo em criar os scripts automatizados de sistema, possui uma dependência de vários fatores que podem estar diretamente relacionado com os vários formatos de equipes de desenvolvimento. Assim, não existe uma resposta única de qual momento adequado para iniciar a automação de teste de sistema, que consiga ser adotada em todos as equipes de desenvolvimento. A partir disso, como esta dissertação é uma teoria fundamentada, identificamos todos os pontos dessa teoria que precisam ser confirmados ou rejeitados em avaliações futuras. Logo, pressupomos conclusões como:

- A participação constante do cliente e a ausência de gerenciamento formal dos requisitos e solicitações de mudanças impedem a criação e/ou manutenção dos scripts de teste de sistema. Logo, a constante participação do cliente no processo de desenvolvimento e a ausência de formalidade no gerenciamento de requisitos influenciam no início da criação dos scripts de teste de sistema em ambientes ágeis, quando comparado com o cliente menos participativo? Provavelmente sim, pois as funcionalidades estão mais sensíveis a mudança, uma vez que o cliente quando muito participativo, pode antecipar alterações que poderiam ser identificadas em um momento posterior e assim ser planejada para os Sprints futuros.

- A implementação dos scripts de teste de sistema quando iniciada ao mesmo tempo que os desenvolvedores iniciam a codificação da funcionalidade aumenta o esforço de manutenção. Portanto, os scripts de teste de sistema escritos muito cedo em ambientes ágeis são mais sensíveis a mudança do que se feitos depois de um certo tempo? Provavelmente sim, pois a criação dos scripts dos testes automáticos de sistema para aplicações web, desktop ou dispositivos móveis possui a dependência com os elementos da interface gráfico do usuário e assim, são mais sensíveis a mudança.

- A ausência de profissionais com experiência em automação de teste de sistema, aumenta o esforço na criação/manutenção e retarda o início da criação dos scripts automatizados. Portanto, em ambientes ágeis, equipe de qualidade sem profissionais experientes em automação de teste de sistema posterga a decisão da automação? A nossa conclusão inicial seria: sim, pois normalmente as pessoas da equipe de qualidade não

gostam de programar e não adotam boas práticas de codificação para os scripts automáticos de testes de sistema, mesmo conhecendo os benefícios da automação.

- A pouca documentação de requisitos criada em ambientes ágeis dificulta a realização da análise da cobertura dos scripts de testes de acordo com os requisitos. Por isso, ao adotar documentação de requisitos mais completa (detalhada) diminui o custo de tempo com a criação e manutenção dos scripts de testes de sistema em ambientes ágeis? Provavelmente concluímos que: sim, diminui o custo de tempo com os requisitos estariam mais definidos, uma vez que existe o detalhamento com as regras que devem ser tratadas e o funcionamento adequado dos requisitos, o analista de teste saberá e terá mais confiança da forma que será validado o requisito, para garantir a satisfação do cliente.

Esses pontos são oportunidades de investigação futura que resultam da nossa teoria.

6.2 Trabalhos Futuros

Com a conclusão deste trabalho, vislumbra-se um conjunto de possíveis trabalhos futuros para continuação do mesmo. Dentre eles é possível destacar:

- A realização de um processo de validação mais abrangente, ou seja, a realização de mais de entrevistas. Assim, poderíamos obter uma variação mais completa dos cenários de desenvolvimento, assim como descobrir novas boas práticas para automação de testes de sistema em cenários de desenvolvimento que adotam práticas ágeis.

- A realização de um processo de validação quantitativa. Essa validação poderia ser realizada através de questionários, que são propagados mais rapidamente em grupos de e-mails e redes sociais; possuiria o objetivo de confrontar os dados analisados qualitativamente.

- A realização de um processo de validação prático, ou seja, as práticas que foram analisadas qualitativamente e quantitativamente seriam introduzidas em vários ambientes de desenvolvimento com práticas ágeis distintas e/ou similares. Assim, esta validação possuiria o objetivo de validar a correlação das práticas e ambientes.

Bibliografia

[1] Emil Alégroth, Robert Feldt, and Pirjo Kolström. Maintenance of automated test suites in industry: An empirical study on visual gui testing. *Information and Software Technology*, 73:66–80, 2016.

[2] Sabina Amaricai and Radu Constantinescu. Designing a software test automation framework. *Informatica Economica*, 18(1):152, 2014.

[3] Scott W Ambler. Ibm agility@ scale™: become as agile as you can be. *IBM Global Services, Somers, NY*, 2010.

[4] José Aparecido. Teste de integração na prática, mar 2017.

[5] James Bach. Exploratory testing explained, 2003.

[6] Maria Aparecida Baggio and Alacoque Lorenzini Erdmann. Teoria fundamentada nos dados ou grounded theory e o uso na investigação em enfermagem no brasil. *Rev Enferm Referência*, 3(3):177–85, 2011.

[7] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[8] Paulo Cheque Bernardo and Fabio Kon. A importância dos testes automatizados. *Engenharia de Software Magazine*, 1(3):54–57, 2008.

[9] Mariana Zanuzzio Blanco. Documentação de teste baseado na norma ieee 829-estudo de caso:"sistema de apoio a tomada de decisão". *Revista TIS*, 1(1), 2012.

[10] Ted Byrt, Janet Bishop, and John B Carlin. Bias, prevalence and kappa. *Journal of clinical epidemiology*, 46(5):423–429, 1993.

[11] Josiane Aparecida Cardoso. Um metodo de testes de integração para sistemas baseados em componentes. 2006.

[12] SH de B Cassiani, Maria Helena Larcher Caliri, and Nilza Teresa Rotter Pelá. A teoria fundamentada nos dados como abordagem da pesquisa interpretativa. *Rev Latino-am Enfermagem*, 4(3):75–88, 1996.

[13] Suranjan Chakraborty and Josh Dehlinger. Applying the grounded theory method to derive enterprise system requirements. In *Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD'09. 10th ACIS International Conference on*, pages 333–338. IEEE, 2009.

[14] Eliane Figueiredo Collins et al. Software test automation practices in agile development environment: An industry experience report. In *Proceedings of the 7th International Workshop on Automation of Software Test*, pages 57–63. IEEE Press, 2012.

[15] Gustavo Emanuel Pinto de Moura Coutinho. O papel das ferramentas de automação de testes funcionais em contexto de projetos dinâmicos e complexos. Master's thesis, 2016.

[16] Carlton A Crabtree, A Gunes Koru, Carolyn Seaman, and Hakan Erdogmus. An empirical characterization of scientific software development projects according to the boehm and turner model: A progress report. In *Software Engineering for Computational Science and Engineering, 2009. SECSE'09. ICSE Workshop on*, pages 22–27. IEEE, 2009.

[17] John W Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2013.

[18] Lisa Crispin and Janet Gregory. *Agile testing: A practical guide for testers and agile teams*. Pearson Education, 2009.

[19] Ubiratan D'Ambrosio. Pesquisa qualitativa em educação matemática. *São Paulo*, 2004.

[20] Knut De Swert. Calculating inter-coder reliability in media content analysis using krippendorff's alpha. *Center for Politics and Communication*, pages 1–15, 2012.

[21] Anca Deak and Tor Stålhane. Organization of testing activities in norwegian software companies. In *Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on*, pages 102–107. IEEE, 2013.

[22] E. Dustin, J. Rashka, and l J. Pau. Automated software testing: introduction management, and performance. In *In Addison-Wesley*, Boston, 1999.

[23] Monique F da Silva and Autran G Moreno. Automaçao em testes ágeis. *Revista de Sistemas e Computação-RSC*, 1(2), 2012.

[24] M. Fewster. Common mistakes in test automation. In *In Fall test automation conference*, Boston, 2001.

[25] Brian Fitzgerald, Klaas-Jan Stol, Ryan O'Sullivan, and Donal O'Brien. Scaling agile methods to regulated environments: An industry case study. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 863–872. IEEE, 2013.

[26] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, page 122, 2006.

[27] Suely Fragoso, Raquel Recuero, and Adriana Amaral. Métodos de pesquisa para internet. *Porto Alegre: Sulina*, 1, 2011.

[28] Deen G Freelon. Recal: Intercoder reliability calculation as a web service. *International Journal of Internet Science*, 5(1):20–33, 2010.

[29] Kelley Cristine Gonçalves Dias Gasque. Teoria fundamentada: nova perspectiva à pesquisa exploratória. 2007.

[30] Barney G Glaser and Anselm L Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Transaction Publishers, 2009.

[31] Arilda Schmidt Godoy. Pesquisa qualitativa: tipos fundamentais. *Revista de Administração de empresas*, 35(3):20–29, 1995.

[32] PAJ Gore, HE Tinsley, and SD Brown. Handbook of applied multivariate statistics and mathematical modeling, 2000.

[33] Dorothy Graham and Mark Fewster. Software test automation: Effective use of test execution tools, 1999.

[34] Michaela Greiler, Arie van Deursen, and Margaret-Anne Storey. Test confessions: a study of testing practices for plug-in systems. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 244–254. IEEE, 2012.

[35] Eduardo Habib. Testes ágeis - como os testes serão modificados com o advento das metodologias ágeis. *Revista Engenharia de Software edição*, 27, 2010.

[36] Rashina Hoda, James Noble, and Stuart Marshall. Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering*, 17(6):609–639, 2012.

[37] RJ Hunt. Percent agreement, pearson's correlation, and kappa as measures of inter-examiner reliability. *Journal of Dental Research*, 65(2):128–130, 1986.

[38] Clélia Cândida Abreu Spinardi Jubran. Revisitando a noção de tópico discursivo. *Cadernos de Estudos Lingüísticos*, 48(1), 2011.

[39] Fernando Kamei, Gustavo Pinto, Bruno Cartaxo, and Alexandre Vasconcelos. On the benefits/limitations of agile so ware development: An interview study with brazilian companies. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pages 154–159. ACM, 2017.

[40] Jussi Kasurinen, Ossi Taipale, and Kari Smolander. Analysis of problems in testing practices. In *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific*, pages 309–315. IEEE, 2009.

[41] Jussi Kasurinen, Ossi Taipale, and Kari Smolander. Software test automation in practice: empirical observations. *Advances in Software Engineering*, 2010, 2010.

[42] Richard H Kolbe and Melissa S Burnett. Content-analysis research: An examination of applications with directives for improving research reliability and objectivity. *Journal of consumer research*, 18(2):243–250, 1991.

[43] M Lombard, J Snyder-Duch, and CC Bracken. Practical resources for assessing and reporting intercoder reliability in content analysis research projects. retrieved april 28, 2004, 2004.

[44] M Lüdke and Meda André. Pesquisa em educação: abordagens qualitativas. são paulo: Epu. *Mathematics Teaching in the Middle School*, 12(5):244–250, 1986.

[45] P. C. Macedo, M. R. de Moraes, and R. de C. Catini. Uso de testes automatizados em projetos de software: estudo da aplicação em software comercial. In *In Universitas - Ano 5 - Nº 9*, 2012.

[46] Nayane Maia, Gisele Macedo, Eliane Collins, and Arilo Dias-Neto. Aplicando testes ágeis com equipes distribuídas: Um relato de experiência. *XI SBQS, Fortaleza, Brasil*, pages 365–372, 2012.

[47] José C. Maldonado. Introdução ao teste de software. In *São Carlos: Instituto de Ciências Matemáticas e de Computação*, 2004.

[48] Agile Manifesto. In *agilemanifesto.org*, 2004.

[49] Eduardo José Manzini. Considerações sobre a transcrição de entrevistas. *Técnicas de Pesquisa: planejamento e execução de pesquisas. Amostragens e técnicas de pesquisa. Elaboração, análise e interpretação de dados*, 7, 2008.

[50] T Muller and D Friedenberg. Certified tester foundation level syllabus. *Journal of International Software Testing Qualifications Board*, 2011.

[51] Kimberly A Neuendorf. *The content analysis guidebook*. Sage publications, 2016.

[52] Reza Meimandi Parizi, Taghi Javdani Gandomani, and Mina Ziaei Nafchi. Hidden facilitators of agile transition: Agile coaches and agile champions. In *Software Engineering Conference (MySEC), 2014 8th Malaysian*, pages 246–250. IEEE, 2014.

[53] Flávia Perin. Garantia de qualidade de software com ênfase em testes de software. In *In REFA - Revista Eletrónica da Fatec Americana*, Americana/SP, Brasil, 2006.

[54] Cândida Martins Pinto. A teoria fundamentada como método de pesquisa. *Anais do XII Seminário Internacional de letras. Disponível em:< http://www. unifra. br/eventos/inletras2012/Trabalhos/4415. pdf>. Acesso em*, 18, 2014.

[55] Roel Popping. On agreement indices for nominal data. In *Sociometric research*, pages 90–105. Springer, 1988.

[56] Lutz Prechelt, Holger Schmeisky, and Franz Zieris. Quality experience: a grounded theory of successful agile projects without dedicated testers. In *Proceedings of the 38th International Conference on Software Engineering*, pages 1017–1027. ACM, 2016.

[57] Jane Radatz, Anne Geraci, and Freny Katki. Ieee standard glossary of software engineering terminology. *IEEE Std*, 610121990(121990):3, 1990.

[58] Roland T Rust and Bruce Cooil. Reliability measures for qualitative data: Theory and implications. *Journal of Marketing Research*, pages 1–14, 1994.

[59] Gery W Ryan, H Russell Bernard, N Denzin, and Y Lincoln. Handbook of qualitative research. *Handbook of qualitative research*, 2000.

[60] A. M. Santos, B. F. Karlsson, and A. M. Cavalcante. Uma abordagem empírica para o tratamento de bugs em ambientes ágeis. In *Workshop Brasileiro de Metodos Ageis (WBMA)*, 2011.

[61] Monalisa Sarma and Rajib Mall. Automatic generation of test specifications for coverage of system state transitions. *Information and Software Technology*, 51(2):418–432, 2009.

[62] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[63] David Silverman. *Qualitative research*. Sage, 2016.

[64] Sarbjeet Singh, Gurpreet Singh, and Sukhvinder Singh. Software testing. *International Journal of Advanced Research in Computer Science*, 1(3), 2010.

[65] Michel S. Soares. Comparaçao entre metodologias ágeis e tradicionais para o desenvolvimento de software. In *In www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf.*, 2004.

[66] Adam Solinski and Kai Petersen. Prioritizing agile benefits and limitations in relation to practice usage. *Software Quality Journal*, 24(2):447, 2016.

[67] Ian. Sommerville. Engenharia de software. 8 ed. In *São Paulo: Pearson Addison-Wesley*, 2007.

[68] Andreas Spillner, Tilo Linz, and Hans Schaefer. *Software testing foundations: a study guide for the certified tester exam*. Rocky Nook, Inc., 2014.

[69] Robert E Stake. *Pesquisa qualitativa: estudando como as coisas funcionam*. Penso Editora, 2016.

[70] Phyllis Noerager Stern. Grounded theory methodology: Its uses and processes. *Image*, 12(1):20–23, 1980.

[71] Zeljko Stojanov, Dalibor Dobrilovic, and Vesna Jevtic. Identifying properties of software change request process: Qualitative investigation in very small software companies. In *Intelligent Systems and Informatics (SISY), 2011 IEEE 9th International Symposium on*, pages 47–52. IEEE, 2011.

[72] Anselm Strauss, Juliet Corbin, et al. *Basics of qualitative research*, volume 15. Newbury Park, CA: Sage, 1990.

[73] Anselm L Strauss and Juliet Corbin. *Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada*. Artmed, 2008.

[74] Ossi Taipale, Katja Karhu, and Kari Smolander. Observing software testing practice from the viewpoint of organizations and knowledge management. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 21–30. IEEE, 2007.

[75] Ossi Taipale, Jussi Kasurinen, Katja Karhu, and Kari Smolander. Trade-off between automated and manual software testing. *International Journal of System Assurance Engineering and Management*, 2(2):114–125, 2011.

[76] Massimiliano Tarozzi. O que é a grounded theory? metodologia de pesquisa e de teoria fundamentada nos dados. *Petrópolis, RJ: Vozes*, 2011.

[77] Hea Tinsly and Dj Weiss. Interrater reliability and agreement of subjective judgements. *Journal of Counseling Psychology*, 22:358–376, 1975.

[78] Luciano Augusto Toledo and Guilherme de Farias Shiaishi. Estudo de caso em pesquisas exploratórias qualitativas: um ensaio para a proposta de protocolo do estudo de caso. *Revista da FAE*, 12(1), 2016.

[79] Sulabh Tyagi, Ritu Sibal, and Bharti Suri. Adopting test automation on agile development projects: A grounded theory study of indian software organizations. In *International Conference on Agile Software Development*, pages 184–198. Springer, 2017.

[80] Cathy Urquhart, Hans Lehmann, and Michael D. Myers. Putting the 'theory' back into grounded theory: guidelines for grounded theory studies in information systems. *Information Systems Journal*, 9999(9999), 2009.

[81] Rui Pedro Costa Vieira. *Caso de demonstração de uma framework para automatização do teste de APIs de aplicações SaaS*. PhD thesis, 2015.

[82] James A Whittaker. *Exploratory software testing: tips, tricks, tours, and techniques to guide test design*. Pearson Education, 2009.

[83] Laurie Williams and Alistair Cockburn. Guest editors' introduction: Agile software development: It's about feedback and change. *Computer*, 36(6):39–43, 2003.

[84] Lili Yu, Xin Xu, Chao Liu, and Bing Sheng. Using grounded theory to understand testing engineers' soft skills of third-party software testing centers. In *Software Engineering and Service Science (ICSESS), 2012 IEEE 3rd International Conference on*, pages 403–406. IEEE, 2012.

Apêndice A

Termo de Consentimento

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

INFORMAÇÕES SOBRE A PESQUISA:

Título do Projeto: Um Estudo sobre Automatização de Teste em Ambientes Ágeis
Orientadores Responsáveis: Patrícia Machado e Tiago Massoni
Pesquisadora Responsável: Gabriella Alves
Telefones para contato: (xx) XXXX-XXXX
E-mail para contato: gabriellamayara@copin.ufcg.edu.br

OBJETIVO DA PESQUISA:

O objetivo desta pesquisa é identificar o momento mais adequado para iniciar a automatização dos testes de sistema em ambientes de desenvolvimento ágil, de funcionalidade naturalmente instável, em que frequentemente os casos de teste tornam-se a principal documentação do software. A coleta das informações será realizada através de entrevista por pessoal/e-mail, caso seja via e-mail, iremos realizar uma sequência de trocas de e-mails até o objetivo da entrevista ser atingido. É provável que, futuramente, após a análise das respostas, haja necessidade de um novo contato para alguns esclarecimentos adicionais. As informações coletadas serão utilizadas como dados para uma dissertação de mestrado, no entanto serão resguardadas informações pessoais e profissionais (como nome da empresa ou de projetos).

TERMO DE CONSENTIMENTO:

Se você concordar voluntariamente em participar do estudo acima descrito, como sujeito. **Torna-se necessário a resposta a este e-mail copiando o trecho a seguir:**

"Declaro ter sido devidamente informado e esclarecido pela pesquisadora Gabriella Alves sobre o objetivo da pesquisa, os procedimentos nela envolvidos, assim como a garantia do sigilo de informações pessoais e profissionais envolvidos na minha participação. Foi-me fornecida a oportunidade de fazer perguntas e recebi telefone e endereço de e-mail para entrar em contato, caso tenha dúvidas. Foi-me garantido que não sou obrigado (a) a participar.".

Apêndice B

Roteiro da Entrevista Via E-mail

ROTEIRO DA ENTREVISTA VIA EMAIL

A entrevista é dividida em quatro partes:

1- Experiência, no qual contempla um conjunto de perguntas sobre sua experiência em conhecimento na área de desenvolvimento de software no geral;
2- Trabalho atual ou/e outras experiências, que possui o objetivo de obter informações do processo de desenvolvimento adotado no seu atual projeto.
3- Atividade de Teste de Software que possui o objetivo de obter informações mais especifica da atividade desenvolvida.
4- Uma prática simples, com o objetivo de colocar o participando em um problema real dentro do desenvolvimento de software.

1. Experiência

a. Qual sua experiência em desenvolvimento ágil?
b. Hoje, quais atividades você desempenha na equipe?
c. Qual sua experiência em teste de software ágil?

2. Desenvolvimento do Sistema – Trabalho atual ou outras experiências

a. Em um contexto geral, como funciona o processo de desenvolvimento atual?
b. Qual é a relação do cliente com o processo de desenvolvimento?
c. Quais os princípios utilizados no gerenciamento do controle de versão do software?

3. Atividade de Teste de Software

a. Como você destaca a atividade de teste de software na garantia da qualidade do sistema?
b. Como funciona o processo de criação dos testes manuais e automatizados em nível de sistema?
c. Quais os desafios encontrados durante a criação e execução dos testes manuais e automatizados?
d. Como são tratadas as mudanças/alterações nos testes automatizados já criados?
e. Como é o gerenciamento de impacto das mudanças/alterações dos testes?
f. Quais os critérios para garantir a conformidade dos testes?
g. Na sua experiência, qual momento para automatizar os testes de sistema?

4. Prático – Cadastro de Usuário

Funcionalidade:

O cadastro do usuário deve possuir os seguintes dados: **nome, cpf, cargo, senha e confirmação de senha.** O usuário deve acessar o sistema através do **código gerado automaticamente** na confirmação do cadastro e a senha informada no cadastro.

Cenários de teste:
Existe uma suíte de teste com 20 casos de testes, no qual possui uma boa cobertura da funcionalidade.

Cenário de desenvolvimento:
Estamos próximo de uma entrega parcial, o tempo está curto e após a suíte de teste ser executada (manualmente e automaticamente), o cliente solicita uma mudança na funcionalidade de cadastro do usuário.

Mudança:
O cadastro do usuário deve possuir os seguintes dados: **nome, matrícula, código de acesso, cargo, senha e confirmação de senha.** O usuário deve acessar o sistema através do código de acesso e a senha informada no cadastro.

Pergunta:
Quais os procedimentos adotados com relação a esta mudança?

Apêndice C

Roteiro da Entrevista Via Gravação

ROTEIRO DA ENTREVISTA VIA GRAVAÇÃO

1. **Experiência**
    a. Qual sua experiência no desenvolvimento de software?
        i. Experiência em metodologia ágil;
        ii. Papel na equipe;
    b. Qual sua experiência em teste de software?
        i. Experiência em teste ágil;

2. **Desenvolvimento do Sistema – Trabalho atual ou outras experiências**
    a. Como é o processo de desenvolvimento adotado?
        i. (Nível e tipo de metodologia ágil)
        ii. Tamanho da Equipe;
    b. Como é a relação com o cliente?
        i. Frequência da iteração com o cliente;
        ii. Frequência de mudanças solicitadas
    c. Como funciona o controle de versão do software (criação de *baseline*)?

3. **Atividade de Teste de Software**
    a. Por que a atividade de teste de software é considerada importante na garantia da qualidade do sistema?
        i. Inicio dos testes (manuais e automatizados)
    b. Como você ver a atividade de teste em um processo de desenvolvimento ágil?
    c. Como funciona o processo dos testes manuais e dos testes automatizados?
        i. Experiência em automatização dos testes de sistema;
        ii. Quando inicia os testes?
    d. Qual a ferramentas e tipos de teste utilizado?
        i. Conhecimento da equipe nas ferramentas utilizadas;
        ii. Tipo de teste abordado (aceitação, unitário, regressão, interface);
    e. Quais os desafios mais encontrados no processo de teste (criação e execução)?
        i. Garantia da conformidade dos testes;
        ii. Desafios na automatização dos testes;
        iii. Gerenciamento da criação, alteração e execução dos testes;
    f. Como é garante a conformidade dos testes e o gerenciamento do impacto das mudanças/alterações dos testes?
        i. Surgimento das mudanças/alterações;
        ii. Gerenciamento das mudanças/alterações;
            1. Quais as práticas adotadas após uma solicitação de mudança?
    g. Na sua experiência, qual momento de automatizar os testes de sistema?

4. **Prático – Cadastro de Usuário**

    Funcionalidade: O cadastro do usuário deve possuir os seguintes dados: **nome, cpf, cargo, senha e confirmação de senha.** O usuário deve acessar o sistema através do **código gerado automaticamente** na confirmação do cadastro e a senha informada no cadastro.

Cenários de teste: Existe uma suíte de teste com 20 casos de testes que cobre esta funcionalidade.
Cenário de desenvolvimento: Estamos próximo de uma entrega parcial, o tempo está curto e após a suíte de teste ser executada (manualmente e automaticamente), o cliente solicita que no cadastro do usuário, exista o campo para matrícula e código de acesso, além de excluir o campo CPF.

Mudança: O cadastro do usuário deve possuir os seguintes dados: **nome, matrícula, código de acesso, cargo, senha e confirmação de senha.** O usuário deve acessar o sistema através do código de acesso e a senha informada no cadastro.

Pergunta: O que você faria com relação a esta mudança?

Apêndice D

Arquivos Gerais

Transcrições e arquivo de análise estão disponíveis em: https://goo.gl/13EMLS