UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

SISTEMAS DE INFORMAÇÃO

VICTOR GABRYEL DA SILVA SANTOS

UMA PROPOSTA DE MELHORIA DO PROCESSO DE TESTES COM BASE NO DIAGNÓSTICO DO USO DAS PRÁTICAS AGILE TESTING

Recife

2022

VICTOR GABRYEL DA SILVA SANTOS

UMA PROPOSTA DE MELHORIA DO PROCESSO DE TESTES COM BASE NO DIAGNÓSTICO DO USO DAS PRÁTICAS AGILE TESTING

Trabalho apresentado ao Programa de Graduação em Sistemas de Informação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Alexandre Marcos Lins de Vasconcelos

Recife

2022

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Santos, Victor Gabryel da Silva.
Uma proposta de melhoria do processo de testes com base no diagnóstico do uso de práticas Agile Testing / Victor Gabryel da Silva Santos. - Recife, 2022.
77 : il., tab.

Orientador(a): Alexandre Marcos Lins de Vasconcelos
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2022.
Inclui referências, apêndices, anexos.

1. Testes Ágeis. 2. Quadrantes dos Testes Ágeis. 3. Processo de teste. 4. Teste de Software. 5. Qualidade de Software. I. Vasconcelos, Alexandre Marcos Lins de. (Orientação). II. Título.

000 CDD (22.ed.)

VICTOR GABRYEL DA SILVA SANTOS

UMA PROPOSTA DE MELHORIA DO PROCESSO DE TESTES COM BASE NO DIAGNÓSTICO DO USO DAS PRÁTICAS AGILE TESTING

Trabalho apresentado ao Programa de Graduação em Sistemas de Informação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Recife, 20 de Outubro de 2022

BANCA EXAMINADORA

_____

Prof. Alexandre Marcos Lins de Vasconcelos (Orientador)

UNIVERSIDADE FEDERAL DE PERNAMBUCO

_____

Prof. Juliano Manabu Iyoda (2º membro da banca)

UNIVERSIDADE FEDERAL DE PERNAMBUCO

Agradecimentos

É com muita alegria que finalizo essa jornada. Após muitos anos de graduação, a universidade e o Centro de Informática me abriram diversas portas. Obrigado Deus por tudo que vivi nesses últimos 5 anos.

De início, gostaria de agradecer a minha família. Sou muito grato ao meu pai Pedro Santos e à minha mãe Shirley da Silva por terem feito todos esforços possíveis para me garantir uma educação de qualidade. Graças a eles sou a pessoa que sou, cheio de sonhos e força de vontade para realizá-los. Tenho a agradecer também a minha companheira Clara Coelho, por sempre me motivar e estar ao meu lado durante todos esses anos. Durante o meu trabalho de graduação, Clara foi uma pessoa que buscou me ajudar, me apoiando emocionalmente e sempre me dando confiança para finalizar esse trabalho. A minha família é extremamente especial para mim .

O CIn me trouxe grandes amizades que levarei para o resto da minha vida, onde essas pessoas foram importantes tanto para nas atividades acadêmicas e projetos mas também na vida pessoal. Ao professor Alexandre Vasconcelos, toda minha gratidão por me orientar neste trabalho! Você foi um excelente orientador, sempre se mostrou à disposição para me ajudar. Obrigado por seu tempo dedicado nesse projeto!

Por fim, quero agradecer ao CESAR! O trabalho foi possível ser realizado graças a todos entrevistados. Ao meu time, meus sinceros agradecimentos por serem tão unidos e pela nossa amizade que vou levar pra sempre.

RESUMO

Nos últimos anos, devido ao crescimento da competitividade no mercado de software, o uso de metodologias ágeis vem se tornando cada vez mais comum em projetos de desenvolvimento. Com isso, a qualidade do produto e processo estão entre as principais características necessárias para o seu sucesso. Nesse contexto, as atividades de testes são fundamentais para validar e garantir a qualidade de um software. Ressalta-se que sua ausência pode acarretar em problemas desde a fase inicial do processo de desenvolvimento até a sua utilização pelos usuários finais.

A partir disso, foi realizado um mapeamento sistemático da literatura, para listar as práticas de teste de software, com base nos Quadrantes de Testes Ágil mais utilizadas desde o surgimento desse modelo. Em seguida, será executada uma pesquisa em campo no modelo de entrevista em um projeto do CESAR, localizado no Porto Digital de Recife. Dessa forma, o trabalho realizou um diagnóstico do grupo analisado e assim propôs melhorias dentro do processo de teste do time, baseando-se nas práticas levantadas na literatura.

Palavras-chave: teste de software, qualidade de software, metodologias ágeis, proposta de melhoria, quadrantes de testes ágeis.

ABSTRACT

In recent years, due to the growth of competitiveness in the software market, the use of agile methodologies has become increasingly common in development projects. Thus, the quality of the product and the process are among the main characteristics necessary for its success. In this context, testing activities are fundamental to validate and guarantee the quality of the software. It is noteworthy that its absence can cause problems from the initial phase of the development process to its use by end users.

From this, a systematic mapping of the literature was carried out, to list the software testing practices, based on the Quadrants of Agile Tests most used since the emergence of this model. Then, a field research was carried out using the interview model in a CESAR project, located in Porto Digital in Recife. In this way, this work carried out a diagnosis of the analyzed group and thus proposed improvements within the team's testing process, based on practices raised in the literature.

Keywords: software testing, software quality, agile methodologies, improvement proposal, agile testing quadrants.

Lista de Ilustrações

Lista de Abreviaturas e Siglas

QTA	Quadrantes de Testes Ágeis
ISTQB	International Software Testing Qualifications Board
TDD	Test-Driven Development
ATDD	Acceptance Test-Driven Development
BDD	Behavior Driven Development
UFPE	Universidade Federal de Pernambuco
XP	Extreme Programming
CESAR	Centro de Estudos e Sistemas Avançados do Recife
UAT	User Acceptance Testing
QA	Quality Assurance
BPMN	Business Process Model and Notation

SUMÁRIO

1. INTRODUÇÃO

O mercado competitivo de *softwares* pressiona as empresas a serem mais eficientes e eficazes, com entregas rápidas e com o fornecimento de produtos bem sucedidos. As estratégias de redução de custo do desenvolvimento de *software* são um dos principais diferenciais, permitindo a entrega de softwares mais rápidos e baratos (WATKINS, 2009). Nesse contexto, a atividade de testes é essencial para a validação e garantia de qualidade de um software. A deficiência nesta atividade pode resultar em softwares que não funcionam como o esperado, acarretando em diversos problemas como a perda de dinheiro e tempo, o que pode desestabilizar competitivamente uma empresa (ISTQB).

Apesar de normalmente o teste de *software* ser definido como a execução de teste em uma aplicação e a verificação dos seus resultados, diversas atividades são importantes para a sua realização. O processo de teste é composto por diversas atividades, desde o planejamento de testes até a avaliação da sua qualidade (ISTQB). O entendimento do processo e sua aplicação de modo correto é fundamental para a redução de riscos e defeitos do sistema. Em caso da aplicação inadequada desses testes, a garantia de qualidade pode falhar e gerar custos adicionais de esforço e tempo.

Adicionalmente, as empresas buscam adotar as metodologias ágeis tendo como objetivo acelerar entregas e melhorar a qualidade do produto (DIGITAL.AI, 2021). Todavia, reconhece-se que o principal desafio para a sua adesão está relacionado a inconsistências em processo e práticas, como as relacionadas a testes.

1.1 Objetivos

A finalidade deste trabalho é elaborar um diagnóstico de um time inserido em um projeto ágil em relação à sua cultura de *Agile Testing* a partir do modelo de Quadrantes de Testes Ágeis (CRISPIN e GREGORY, 2009). Assim será possível entender melhor o uso das práticas e técnicas de teste de *software* dentro do seu processo de desenvolvimento. Com base nesse levantamento, será possível fazer um paralelo entre o estado da prática e o estado da arte, identificando as principais dificuldades encontradas e propondo melhorias para o processo de teste.

1.2 Metodologia

Para alcançar os objetivos estabelecidos, o trabalho está dividido nas seguintes três etapas:

- Executar uma revisão da literatura para investigar e listar as práticas de teste de software, com base nos Quadrantes de Testes Ágeis.
- Definir e elaborar uma pesquisa qualitativa sobre o entendimento de um time inserido em um projeto ágil em relação às práticas e técnicas de teste ágil extraídas na primeira etapa. Desta forma, será realizado um diagnóstico a partir dessas informações coletadas do grupo analisado.
- Propor melhorias dentro do processo de teste do time, com base nas práticas de teste levantadas na literatura.

1.3 Estrutura do trabalho

A organização do trabalho se dá pela seguinte forma:

- Capítulo 2 – Apresenta conceitos sobre testes de software, metodologias ágeis e as práticas de Testes Ágeis.
- Capítulo 3 – Apresenta o mapeamento sistemático da literatura relativa a testes ágeis. Todo o processo de seleção de estudos é detalhado neste capítulo. Além disso, são apresentados os dados obtidos com a pesquisa.
- Capítulo 4 – Apresenta a pesquisa realizada em um projeto C.E.S.A.R do Porto Digital do Recife e os dados coletados. A partir da análise do resultado da pesquisa foi desenvolvido um diagnóstico de Testes Ágeis.
- Capítulo 5 – Apresenta a proposta de um conjunto de melhorias baseadas no diagnóstico realizado.
- Capítulo 6 – Apresenta as considerações finais da pesquisa e também as recomendações para trabalhos futuros.

2. REFERENCIAL TEÓRICO

2.1 Introdução

O objetivo deste capítulo é descrever os conceitos que fundamentam este trabalho. De início, apresenta-se o conceito de testes de *software* e sua importância para a qualidade de um produto. Em seguida, são apresentadas as principais metodologias ágeis e suas definições. No final do capítulo serão descritas as principais práticas de testes ágeis relativas ao modelo de Quadrantes de Testes Ágeis.

2.2 Teste de Software

Diante do crescimento da dependência da sociedade com relação à tecnologia, empresas vêm cada vez mais buscando desenvolver produtos de software com maior qualidade. Porém, devido à possibilidade da introdução de falhas no software, é de suma importância que os mesmos sejam encontrados antes de sua utilização real. Neste contexto, os testes se apresentam como uma possível solução para tentar garantir a qualidade do software desenvolvido.

A finalidade do teste é mostrar que um determinado software apresenta o comportamento esperado, além de evitar o uso do software com defeitos (SOMMERVILLE, 2011). Ainda de acordo com Sommerville (2011), testar não garante que um sistema é livre de defeitos, mas pode demonstrar que existem problemas. De acordo com o (ISTQB), além de permitir que defeitos sejam detectados e corrigidos de forma antecipada, o teste de software é de suma importância para garantir o atendimento dos requisitos contratuais ou legais. Dessa forma, pode-se dizer que a aplicação de testes em projetos de desenvolvimento de *software* é fundamental para o seu sucesso.

Os testes de *software* podem ter objetivos variados dependendo do contexto em que são aplicados. Alguns objetivos típicos dos testes são:

- A prevenção de defeitos a partir da avaliação dos produtos;
- Verificação do cumprimento de requisitos;
- Avaliação do funcionamento correto da aplicação de acordo com as expectativas do cliente e dos usuários;
- Garantia da qualidade do *software* reduzindo defeitos e falhas de sistema;

- Geração da confiança ao time e cliente acerca da qualidade do produto entregue.

2.3 Metodologias Ágeis

Em meados dos anos de 1980 à 1990, na Engenharia de Software, o desenvolvimento de software seguia um processo rigoroso e controlado. Essa definição foi estabelecida pela comunidade de Engenharia de Software, responsável por sistemas de software de grande porte, como sistemas aeroespaciais (SOMMERVILLE, 2011). Contudo, segundo Sommerville (2011), por ser uma abordagem pesada, o desenvolvimento de sistemas corporativos menos críticos tinham uma grande sobrecarga. O gasto excessivo de tempo com o planejamento antes do desenvolvimento do sistema era um dos principais problemas criticados.

A partir da necessidade de um processo mais rápido, um grande grupo de desenvolvedores, na década de 1990, propôs o Manifesto Ágil (SOMMERVILLE, 2011). De acordo com Melo et al. (2013), o Manifesto Ágil (FOWLER e HIGHSMITH, 2001) surgiu em 2001, criado por um grupo de 17 desenvolvedores influentes na área de Engenharia de Software. O manifesto teve como proposta os seguintes valores:

- Indivíduos e interações mais do que processos e ferramentas;
- *Software* em funcionamento mais do que documentação abrangente;
- Colaboração do cliente mais do que negociação de contrato;
- Respostas a mudanças mais do que seguir um plano.

Além disso, existem 12 princípios estabelecidos pelo manifesto que os Métodos Ágeis devem seguir:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência na menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- A contínua atenção à excelência técnica e bom design aumenta a agilidade.
- A simplicidade é essencial.
- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

De acordo com Leite (2022), com o decorrer dos anos após o lançamento do Manifesto Ágil (FOWLER e HIGHSMITH, 2001), cerca de 50 métodos ágeis foram criados, formados por diferentes conjuntos de técnicas e estratégias, suportando os times de desenvolvimento. De acordo com o *15th Annual State of Agile Report* (DIGITAL.AI, 2021), as metodologias Scrum, Kanban e XP estão entre as mais relevantes em projetos ágeis e por isso serão descritas a seguir.

2.3.1 Scrum

O Scrum é um método ágil com base no gerenciamento do desenvolvimento iterativo. As abordagens técnicas de Engenharia de Software ágil não são seu foco, não prescrevendo o uso de práticas de programação (SOMMERVILLE, 2011). Para Watkins (2009), esse método tem como princípio o reconhecimento de que usuários e clientes podem mudar com uma certa frequência suas necessidades e opiniões acerca do que está sendo desenvolvido. A partir desse contexto, o foco do desenvolvimento deve estar relacionado ao que realmente importa: entregas rápidas de acordo com as novas necessidades.

De acordo com Sommerville (2011), o Scrum está dividido em três fases, sendo elas respectivamente:

- O planejamento geral que define os objetivos gerais do projeto e a arquitetura do sistema.

- Os ciclos de *sprint,* onde é gerado o incremento do sistema a cada ciclo. Essa fase é a mais inovadora e principal característica desse framework, já que ao fim de cada ciclo uma parte do sistema é entregue aos *stakeholders.*

- Encerramento do projeto. Nesta fase é entregue a documentação exigida, manuais de ajuda ao usuário e é feita a avaliação das lições aprendidas durante todo o processo do projeto.

Para Sommerville (2011), o Scrum apresenta práticas importantes na colaboração do time em decisões. As reuniões diárias, uma das práticas do Scrum, devem ser realizadas com todo o time para compartilhar o progresso das atividades, discussões de problemas e atividades planejadas. O termo "*stand-up*" seria a ideia de realizá-las em pé, de modo a encurtar o tempo e manter o time focado. Essa prática de *daily standups* é utilizada por 87% dos projetos ágeis (DIGITAL.AI, 2021). Ainda de acordo com a pesquisa realizada pela Digital.ai (2021), o Scrum é a metodologia ágil mais popular, possuindo algumas variações como o ScrumBan ou Scrum/XP para complementá-la com técnicas específicas de programação.

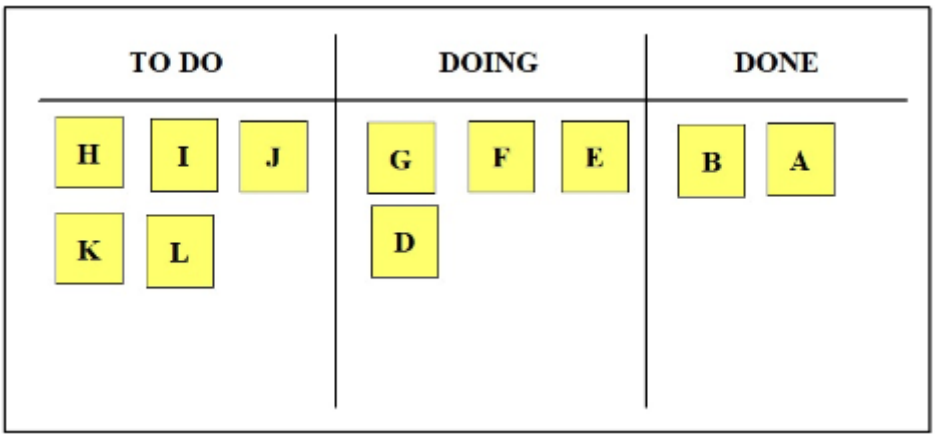
2.3.2 KANBAN

O Kanban é uma metodologia que foi introduzida pela Toyota em 1962 para solucionar o problema de desperdício de recursos. O seu objetivo era otimizar o fluxo de materiais para as suas montadoras de automóveis, se beneficiando do uso mínimo do recurso (WAKODE et al., 2015). O método ágil Kanban permite que os membros de um time possam gastar esforços em uma atividade específica e ao concluí-la é possível puxar outra tarefa. Com isso, o seu uso é importante para o gerenciamento de fluxo de trabalho em projetos de desenvolvimento de software (ALAIDAROS et al., 2021).

De acordo com Leite (2021), a metodologia Kanban tem como a principal ferramenta o quadro Kanban. A responsabilidade dessa ferramenta é deixar visível todo o fluxo de atividades para o time. O quadro Kanban é a ferramenta ágil mais utilizada, presente em 77%

dos projetos (DIGITAL.AI 2021). De acordo com a Figura 1, é possível notar que o quadro é dividido em três colunas, sendo cada coluna o estado atual de uma tarefa. A coluna à esquerda contém todas as atividades que estão em espera para serem desenvolvidas. Ao centro, se encontram atividades que estão em percurso, sendo esperado que apenas um integrante possua uma atividade nessa fase. Por último, à direita é apresentada as atividades que já foram concluídas. Durante todo o processo de desenvolvimento é esperado que cada atividade inicie na coluna mais à esquerda e finalize na coluna mais à direita.

Figura 1 - Quadro Kanban simplificado



Fonte: (ALAIDAROS et al., 2021)

2.3.3 Extreme Programming (XP)

A Extreme Programming, também conhecida como XP, surgiu quando Kent Beck, um profissional de Engenharia de Software, acreditou que existia a possibilidade de simplificar e trazer maior eficiência ao processo de desenvolvimento de software (WATKINS, 2009). Segundo Beck (2000), XP é uma metodologia leve, eficiente, flexível e de baixo risco para atender necessidades específicas de desenvolvimento de softwares por equipes pequenas que possuem dificuldades com a imprecisão e mudanças dos requisitos. De acordo com Watkins (2009), a importância de testes eficazes é enfatizada por essa metodologia, através da recomendação de criação de testes antes do desenvolvimento da funcionalidade.

Essa metodologia possui uma série de práticas que estão alinhadas aos princípios do manifesto ágil, como descritas a seguir:

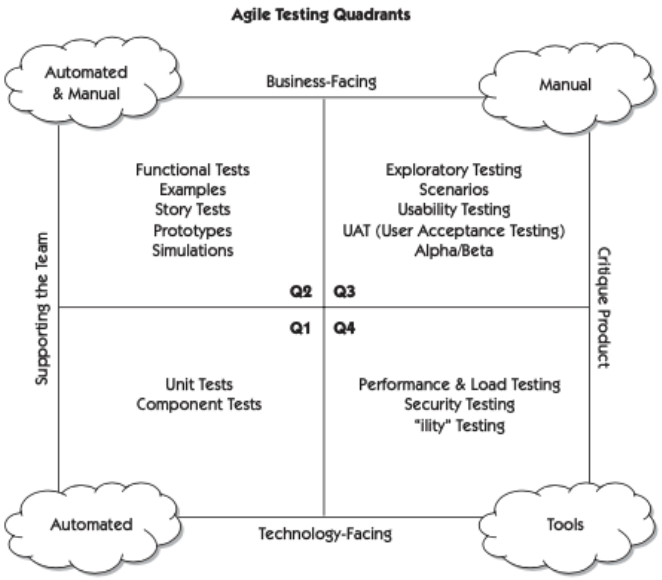
- Planejamento incremental: as estórias devem ser incluídas em uma release de acordo com o tempo para seu desenvolvimento e sua prioridade. A partir desse planejamento, elas são divididas em tarefas a serem implementadas;
- Pequenos *releases*: desenvolver um conjunto mínimo de features que agregam valor e assim liberar, com frequência, versões que entreguem funcionalidades adicionadas de forma gradual;
- Projeto Simples: realizado para atender as necessidades atuais e nada além disso;
- Desenvolvimento do teste primeiro: o time de desenvolvimento escreve primeiro os testes usando algum *framework* antes de iniciar a implementação da funcionalidade.
- Refatoração: manter o código simples a partir do desenvolvimento de melhorias de forma contínua;
- Programação em pares: uma atividade de desenvolvimento é realizada por dois programadores com a finalidade de solucionar o problema em conjunto;
- Propriedade coletiva do código: todos os membros do time de desenvolvimento devem ter responsabilidade por todo o código, sendo assim mudanças podem ser feitas por qualquer pessoa do time;
- Integração Contínua: prática de integração das alterações código desenvolvido pelos programados de forma frequente;
- Ritmo sustentável: as horas-extras devem ser minimizadas pois podem gerar problemas na produtividade;
- Cliente no local: é necessário que o time tenha acesso frequente ao cliente e assim possa levantar requisitos, validações e mudanças.

2.4 Testes Ágeis

No ambiente de desenvolvimento ágil, é notório que os métodos ágeis buscaram propor práticas de desenvolvimento com o cuidado na qualidade do *software*. Como exemplo, o desenvolvimento orientado a testes é uma das principais práticas ágeis, via a escrita de pequenos testes com base no requisito, para desenvolver uma funcionalidade que entregue o resultado que o testes espera.

O modelo de Quadrantes de Testes Ágeis (QTA) criado por Crispin e Gregory (2009) é uma matriz com o propósito de separar os testes em diferentes categorias de qualidade: foco no negócio, foco da tecnologia, foco no suporte ao time e foco na crítica do produto.

Figura 2 - Quadrantes de Testes Ágeis



Fonte: (CRISPIN e GREGORY, 2009)

Como visto na Figura 2, os Quadrantes de Testes Ágeis são divididos em quatro partes, a saber:

- **Primeiro Quadrante de Testes Ágeis (Q1)**: testes direcionados à tecnologia com foco no suporte ao time.
- **Segundo Quadrante de Testes Ágeis (Q2)**: testes direcionados aos negócios com foco no suporte ao time.
- **Terceiro Quadrante de Testes Ágeis (Q3)**: testes direcionados aos negócios com foco na crítica do produto.
- **Quarto Quadrante de Testes Ágeis (Q4)**: testes direcionados à tecnologia com foco na crítica do produto.

2.4.1 Primeiro Quadrante

O propósito do quadrante Q1 é ajudar o time de desenvolvimento sobre a especificação de requisitos, garantindo que o desenvolvedor tenha o entendimento do que o código deve fazer (CRISPIN e GREGORY, 2009). Além disso, as práticas relativas a esse quadrante devem avaliar a infraestrutura do software para assim certificar a sua qualidade interna. Algumas práticas relacionadas a este quadrante são:

2.4.1.1 Test-Driven Development

O Test-Driven Development (TDD) é uma abordagem, apresentada no Extreme Programming, que tem como objetivo a alternancia de testes e desenvolvimento de código durante o processo de desenvolvimento (SOMMERVILLE, 2011). Tal prática colabora na compreensão, por parte do desenvolvedor, do que deve ser implementado.

Segundo Sommerville (2011), todo o processo do TDD é dividido em cinco etapas. Inicialmente é necessário identificar qual será o incremento de código que será desenvolvido, sendo importante ser o menor possível. Após isso, deve-se criar um teste automatizado para essa funcionalidade que será executado em todo momento que o sistema for executado. Na terceira etapa o teste é executado, tendo a falha como resultado por não existir nenhum código implementado. Na próxima etapa, o desenvolvedor deve implementar a funcionalidade e assim executar o teste para verificar se passou. Em caso de falha na execução do teste, a refatoração deve ser realizada para que todos os testes passem. Ao fim, com o sucesso da execução dos testes, o desenvolvedor deve prosseguir para a próxima funcionalidade.

2.4.1.2 Testes Unitários

Os testes unitários têm como objetivo avaliar se os requisitos definidos são atendidos a partir de testes executados pelo desenvolvedor. Para isso, é necessário a execução de testes individuais de pequenos componentes de um software (Runeson, 2006). De acordo com Sommerville (2011), é essencial a automação dos testes unitários com o uso de *frameworks*. Assim, é possível a execução rápida dos testes com uma maior frequência, garantindo que unidades do código não foram afetadas com alterações no código.

2.4.1.3 Programação em Par

A programação em par é uma das práticas encontradas na metodologia ágil Extreme Programming. Seu objetivo é a escrita de código por dois desenvolvedores em um mesmo ambiente de trabalho, lado a lado (WATKINS, 2009). Por definição, existem dois papéis envolvidos no uso dessa prática. O papel do primeiro desenvolvedor é responsável por escrever o código e executá-lo. O segundo papel está relacionado com o acompanhamento e ajuda ao primeiro papel, revisando e sugerindo ideias (LIMA, 2013).

Para Lima (2013), essa prática possui alguns benefícios esperados quando aplicada no processo de desenvolvimento, tais como:

- Código menos defeituoso;
- Maior produtividade a longo prazo;
- Desenvolvimento de soluções mais complexas em menos tempo;
- Meio de aprendizado e alinhamento de conhecimentos.

2.4.1.4 Refatoração

Também presente na lista de práticas do XP, a refatoração é uma das práticas de desenvolvimento. Refatoração tem como finalidade fazer melhorias no código de uma *feature* existente a fim de simplificar o programa (BECK, 2000). Para Sommerville (2011), a refatoração é fundamental para a viabilidade do desenvolvimento de um *software*. Sem a refatoração, o custo e a complexidade de fazer mudanças no código seriam bem altos. Alguns maus cheiros comuns na Engenharia de Software devem ser minimizados com o uso dessa prática, tais como:

- Código duplicado;
- Métodos longos;
- Reincidência de dados no mesmo programa ;
- Código não utilizado.

2.4.2 Segundo Quadrante

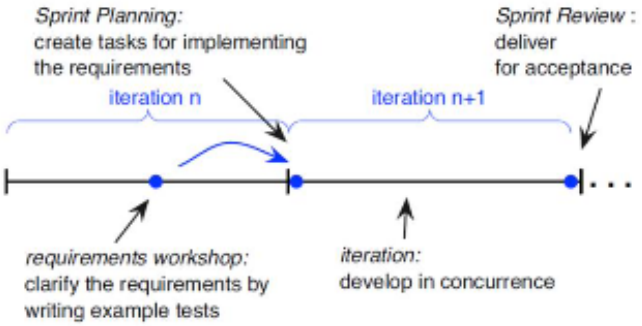
Os testes relativos a esse quadrante tem como finalidade verificar a qualidade externa do produto.

2.4.2.1 Desenvolvimento Orientado a Testes de Aceitação

A Acceptance Test-Driven Development (ATDD) tem como propósito, com base nos princípios ágeis, o desenvolvimento pelo cliente, desenvolvedor e testador de novos testes de aceitação. A estratégia formada por essa tríade de papéis deve ajudar na definição de novos requisitos antes de implementá-los (Hoffman et al., 2014).

Na Figura 3 observa-se que, ao ser usado com a metodologia ágil Scrum, são desenvolvidos testes de aceitação pela tríade na metade da primeira iteração, antes do planejamento da próxima *sprint*. A partir disso é possível desenvolver todos os requisitos descobertos e ao final da iteração uma versão do sistema deve ser liberada para aceitação do usuário.

Figura 3 - ATDD usado com a metodologia Scrum



Fonte: (HOFFMAN et al., 2014)

Para Swaminathan et al., essa prática que é comumente usada em times ágeis possui alguns benefícios, sendo esses alguns dos principais:

- Maior colaboração entre cliente, desenvolvedor e testador;
- Envolvimento do time de testes já no estágio de desenvolvimento;
- Encontrar *bugs* antecipadamente;
- Reduzir custos melhorando a produtividade e qualidade.

2.4.2.2 Testes Automáticos

Para Crispin e Gregory (2009), a automação de testes tem como objetivo a execução de atividades de testes sem a necessidade do esforço humano. Com isso, a diminuição de trabalhos repetitivos e do tempo de validação do sistema é um dos benefícios advindos da sua aplicação em um projeto de software.

Existem diversos testes ou processos que podem ser automatizados em um projeto

ágil. Segundo Crispin e Gregory (2009), o processo de deployment automatizado que consiste em Integração Contínua e criação de builds pode acelerar a velocidade com que a aplicação é testada reduzindo erros.

2.4.2.3 Testes Funcionais

Os testes funcionais avaliam as funcionalidades que um *software* deve executar (ISTQB). Na Engenharia de Software as funções são o que regra um sistema, definido seu comportamento. Essa avaliação das funcionalidades tem como objetivo detectar se algo não está funcionando de acordo com os requisitos definidos. Assim, problemas como uma implementação que não está de acordo com as necessidades do usuário podem ser encontrados.

2.4.3 Terceiro Quadrante

O Terceiro Quadrante de Testes Ágeis busca criticar o produto desenvolvido, verificando se o comportamento, aparência ou fluxo, são satisfatórios para os usuários finais. Portanto, algumas práticas relacionadas a este quadrante são:

2.4.3.1 Testes de Usabilidade

Os testes de usabilidade têm como objetivo identificar o nível de qualidade de um produto em relação à usabilidade de interface de acordo com o que é relatado pelos usuários (FERREIRA, 2002). Diante desses testes, é possível avaliar e entender a criticidade dos problemas reportados, tendo como parâmetro o grau de insatisfação do usuário. A partir disso, essa avaliação deve resultar na priorização desses problemas de usabilidade para satisfazer as necessidades do usuário.

Para Nielsen (1994) a qualidade da usabilidade é perceptível a partir desses princípios:

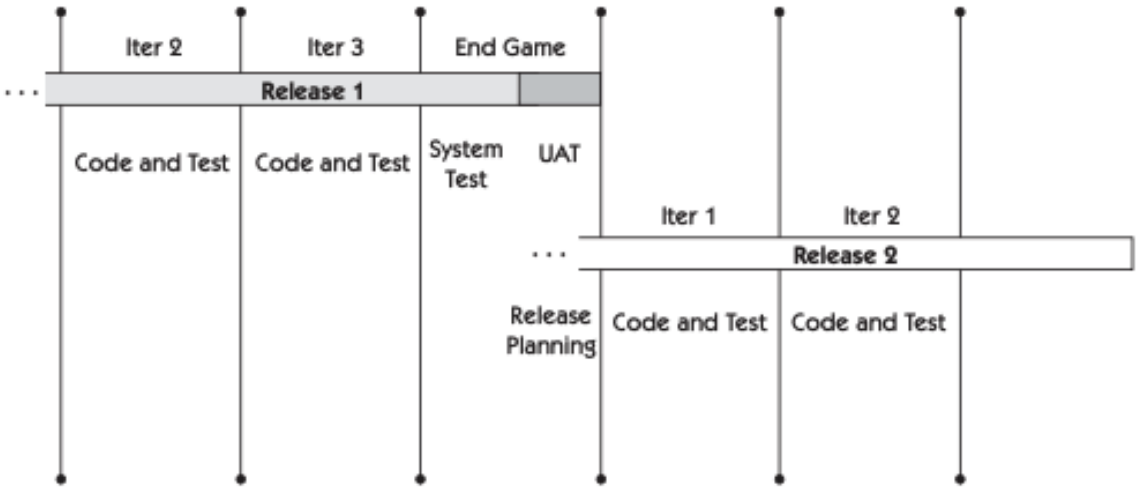
- Facilidade de aprendizado;
- Facilidade de memorização de tarefas no caso de uso intermitente;
- Produtividade dos usuários na execução de tarefas;
- Prevenção, visando a redução de erros por parte do usuário;

- Satisfação subjetiva do usuário.

2.4.3.2 Testes de Aceitação do Usuário

O User Acceptance Testing (UAT) é um exemplo de teste de software realizado pelo usuário, com o propósito de gerar feedbacks da sua aprovação ou rejeição do produto (CRISPIN e GREGORY 2009). Para produtos de grande porte ou de uso interno na empresa, sua aplicação pode ser necessária para entender sua aceitação diante das pessoas que irão utilizá-la. Para a aplicação desse tipo de teste, é necessário o projeto ter acesso a alguns usuários finais e ao cliente para assim poder mediar como devem ser executados os testes e coletar o relatório final da crítica ao produto. Como demonstrado na Figura 4, durante a fase final de desenvolvimento é importante executar os testes de aceitação do usuário. Assim, é possível entender se o funcionamento das *features* estão de acordo com o esperado e também poder trabalhar em melhorias pontuais antes de seu lançamento.

Figura 4 - Cronograma de lançamento com UAT



Fonte: (CRISPIN e GREGORY 2009)

2.4.3.3 Testes Exploratórios

Os testes exploratórios se resumem em aprendizagem, projeto e execução de testes realizados de maneira simultânea (BACH 2003). De acordo com Castro (2018), essa abordagem pode trazer maior produtividade em projetos com testadores já adaptados a todo

fluxo e requisitos do sistema. Atualmente nos testes exploratórios são encontradas duas abordagens principais. A primeira tem como objetivo a execução de teste seguindo um script. Na segunda abordagem esta atividade deve ser realizada de forma livre, apenas com o intuito de encontrar problemas comuns e problemas de comportamento.

Para Castro (2018), os testes exploratórios podem ser aplicados para trazer benefícios quando:

- Existe a necessidade de feedback a curto prazo;
- Não existe tempo suficiente para execução de um ciclo completo;
- Há necessidade de testes com visão do usuário final;
- É necessário executar testes vagos ou imprecisos;
- Documentação de requisitos não é disponibilizada.

2.4.4 Quarto Quadrante

O último quadrante do modelo de Quadrantes de Testes Ágeis agrupa técnicas e práticas de testes que procuram maneiras de criticar o produto. Esse quadrante possui uma visão voltada à tecnologia que é usada, sendo representados pelos testes não funcionais.

2.4.4.1 Testes de Desempenho

Os testes de desempenho são propostos para entender o impacto no sistema com aumento de carga, buscando encontrar os problemas de processamento de informações (SOMMERVILLE, 2011). Ainda conforme Sommerville (2011), para sistemas baseados em redes de processadores, problemas como o baixo desempenho à medida que são sobrecarregados são comuns e esperados. Nesse tipo de aplicações, os testes de desempenho são relevantes para detectar onde possíveis gargalos de processamento são introduzidos. Com essas informações coletadas, é possível ter o controle do sistema e assim tomar medidas preventivas.

Segundo Molyneaux (2009), são alguns termos relacionados aos testes de performance:

- Teste de estresse: mensurar o desempenho do software em cenários de limite de capacidade da infraestrutura;
- Teste de carga: validar as características como disponibilidade, simultaneidade, tempo de resposta de um sistema são atendidas;

2.4.4.2 Testes de Segurança

Os testes de segurança têm como um conceito básico verificar a qualidade da implementação dos requisitos de segurança (FELDERER et al., 2016). Para Whitman (2011), em uma organização é necessário ter algumas medidas de segurança em relação a informações para garantir sua confidencialidade, integridade e confiabilidade.

De acordo com esse Felderer et al. (2016), são apresentadas as principais técnicas de teste para aumentar a segurança de um sistema de tecnologia:

- Testes baseados em Modelos: tem como importância melhorar a documentação de casos de testes, gerar automaticamente testes úteis e o gerenciamento a cobertura de testes ;
- Análise estática e testes baseados em Código: a análise tem sua importância de detectar possíveis vulnerabilidades no estágio inicial do desenvolvimento;
- Testes de penetração e análise dinâmica: utiliza-se de testes de caixa preta para realizar tentativas de penetração durante a execução do sistema, reproduzindo a mentalidade de um invasor;
- Testes de regressão de segurança: garantir que a evolução do código não afetou a segurança do sistema.

3. MAPEAMENTO SISTEMÁTICO DA LITERATURA

3.1 Introdução

O objetivo deste capítulo é realizar um mapeamento sistemático da literatura relacionado com as principais práticas de testes ágeis com base no modelo dos Quadrantes de Testes Ágeis (CRISPIN e GREGORY, 2009).

3.2 Perguntas da Pesquisa

De acordo com Kitchenham e Charters (2007), a especificação das perguntas para uma revisão é a etapa mais importante. No escopo deste trabalho, a Pergunta Primária de Pesquisa (PPP) é: **Quais as técnicas/práticas de testes ágeis são mais relevantes no contexto do modelo dos Quadrantes de testes ágeis?**

As questões secundárias com o objetivo de orientar a pesquisa são:

- PPS1: Quais as técnicas que se encontram no Primeiro Quadrante de Testes ágeis que são mais relevantes?
- PPS2: Quais as técnicas que se encontram no Segundo Quadrante de Testes ágeis que são mais relevantes?
- PPS3: Quais as técnicas que se encontram no Terceiro Quadrante de Testes ágeis que são mais relevantes?
- PPS4: Quais as técnicas que se encontram no Quarto Quadrante de Testes ágeis que são mais relevantes?

3.3 Estratégia de Busca

A seguir serão apresentados os principais parâmetros e ferramentas utilizadas para responder às perguntas da pesquisa.

3.3.1 Fontes de Busca

Para a coleta dessas técnicas que são de extrema importância para diagnosticar o grupo estudado, foram selecionados 3 repositórios com estudos primários, sendo todos

recomendados por Brereton et al. (2007). As bases utilizadas são: **IEEE**, **ACM Digital Library** e **ScienceDirect**.

3.3.2 Termos de Busca

Tendo como finalidade encontrar a maior quantidade de estudos relacionados com a pergunta central da pesquisa, foi necessário definir palavras chaves e seus sinônimos para que assim fosse possível utilizá-los dentro dos termos de busca. A Tabela 1 reúne todas as palavras-chaves utilizadas e também seus sinônimos.

Tabela 1: Palavras-chave e sinônimos

Palavras-chave	**Sinônimos**
agile testing	agile test
methods	techniques, practices

Fonte: Elaborado pelo autor

3.3.3 String de Busca

Tendo como finalidade conseguir englobar a maior quantidade possível de estudos, foi necessário a definição de uma string de busca, que utiliza operadores lógicos "OR" e "AND". O operador "OR" foi empregado para englobar todos os sinônimos de palavras-chave, já o "AND" foi empregado para garantir a presença das palavras-chave. A string formulada encontra-se na Tabela 2:

Tabela 2: *String* de busca

***String* de busca**
("agile testing" OR "agile test") AND ("practices" OR "methods" OR "techniques")

Fonte: Elaborado pelo autor

3.4 Critério de Seleção dos Estudos

Para Kitchenham e Charters (2007), os critérios de seleção de estudos são necessários para identificar os estudos primários que contenham evidências sobre o objetivo principal da pesquisa, sendo os critérios de inclusão e exclusão uma forma de garantir a redução da probabilidade de viés, sendo esses definidos pré-pesquisa, mas com possibilidade de ajustes. Para esse trabalho, os critérios de inclusão (CI) e critérios de exclusão (CE) foram destrinchados nas tabelas 3 e 4, respectivamente.

3.4.1 Critérios de Inclusão

Tabela 3: Critério de Inclusão

Critério de Inclusão	**Definição**
CI1	Publicações que dissertam sobre Testes Ágeis
CI2	Publicações que dissertam sobre práticas, técnicas ou tipos de testes ágeis
CI3	Trabalhos publicados após a definição dos Quadrantes de Testes Ágeis (2009)
CI4	Documentos possíveis de acessar gratuitamente ou pela rede da UFPE.

Fonte: Elaborado pelo autor

3.4.2 Critérios de Exclusão

Tabela 4: Critério de Exclusão

Critério de Exclusão	**Definição**
CE1	Publicações que não dissertam sobre Testes Ágeis
CE2	Publicações que não dissertam sobre práticas, técnicas ou tipos de testes ágeis
CE3	Trabalhos publicados antes da definição dos Quadrantes de Testes Ágeis (2009)
CE4	Documentos sem acesso gratuito ou pela rede da UFPE.

CE5	Livros, testes, jornais, papers, trabalhos duplicados, dissertações
CE6	Documentos que não foram escritos na língua inglesa

Fonte: Elaborado pelo autor

3.5 Procedimento para a Seleção dos Estudos

A seleção dos estudos foi dividida em 4 fases. Na primeira fase utilizou-se a *string* de busca anteriormente definida para encontrar estudos primários relacionados com as palavras-chave e sinônimos e também foram aplicados os critérios de exclusão para listar todos os artigos. É importante ressaltar que a listagem dos artigos foi realizada por ordem de relevância, por meio da ferramenta de ordenação de cada repositório, já que o objetivo era coletar as técnicas mais relevantes.

Na próxima fase, foi realizada uma leitura do título e resumo de cada estudo coletado na fase anterior, aplicando ambos critérios de seleção. Assim, foi possível fazer uma análise mais genérica e com isso os estudos aprovados seguiram para a próxima fase de análise.

Na terceira fase do processo de seleção, a estratégia foi analisar os campos de introdução e conclusão dos estudos remanescentes. Foram aprovados os estudos que não responderam de forma clara a pergunta central da pesquisa, para que fosse possível fazer uma análise mais detalhada e assim não perder nenhum potencial trabalho. Os artigos que não respeitaram os critérios de seleção foram reprovados e não seguiram para a próxima fase.

Na última fase foi feita uma leitura completa de todos documentos que sobraram após as três primeiras fases de seleção. Com isso foi possível verificar se cada artigo respondia pelo menos uma das questões secundárias, para assim conseguir extrair práticas mapeadas dentro do documento.

3.6 Extração dos Dados

Seguindo as recomendações de Kitchenham e Charters (2007), foram armazenados os dados relevantes para identificação e *status* do processo de seleção, sendo esses dados o título do documento, repositório, nome do autor, além da coluna de resultado foram usados "Aprovado" e "Rejeitado" como *keywords* para definir o status de cada documento por fase. Na última etapa foram coletadas quais práticas eram descritas nos trabalhos com base no

modelo dos Quadrantes de Testes Ágeis. Como *software* para armazenamento dos dados e geração de gráficos foi utilizado o Google Planilhas.

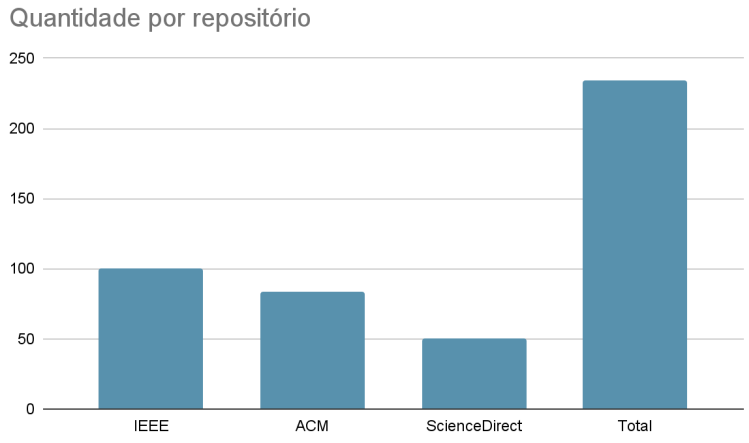
3.7 Resultados

O objetivo dessa seção é elencar todos resultados relevantes obtidos após o processo de seleção de estudos, sendo o principal foco a coleta das práticas mais relevantes de acordo com o mapeamento.

3.7.1 Resultado do Procedimento de Seleção dos Estudos

Com base no processo definido na Seção 3.6 foram selecionados inicialmente 234 estudos. A Figura 5 mostra a quantidade de artigos encontrados por fonte de busca, sendo a IEEE a mais relevante em quantidade de estudos encontrados.

Figura 5: Quantidade de estudos por repositório



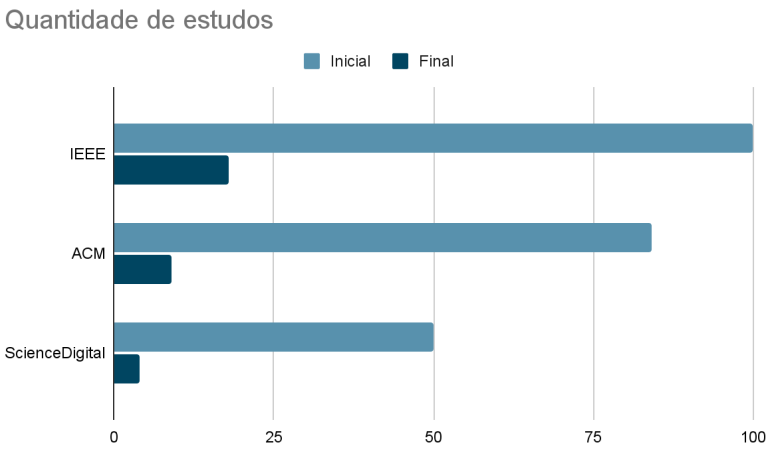
Fonte: Elaborado pelo autor

Após a primeira fase de análise, após a leitura dos títulos e resumos, um total de 95 artigos foram selecionados para a próxima etapa. Em seguida, foi feita a avaliação das seções de introdução e de conclusão que resultou na aprovação de 34 artigos primários.

Como mostra a Figura 6, na última fase, onde o revisor fez uma leitura completa de cada documento, foi possível ter uma quantidade bem reduzida comparada com a coleta inicial. Sendo assim, a coleta das práticas e técnicas de testes ágeis se deu por uma análise de

32 estudos, onde a maior quantidade de documentos está indexada na IEEE. Conforme a conclusão da extração final dos estudos foi viável mapear cada prática referente aos testes ágeis.

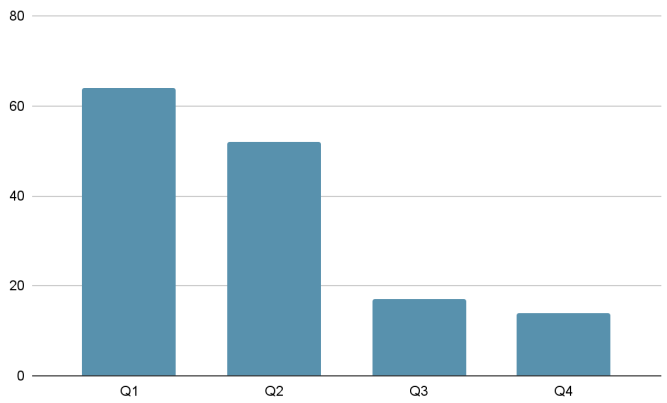
Figura 6: Quantidade de estudos final por repositório



Fonte: Elaborado pelo autor

Na Figura 7 é possível notar que a quantidade foi superior de citações a práticas relacionadas aos quadrantes 1 e 2 dos Quadrantes de Testes Ágeis, quadrantes esses que têm um foco em testes de suporte ao time de desenvolvimento, sendo voltados à tecnologia e aos negócios respectivamente.

Figura 7: Quantidade de práticas por quadrante



Fonte: Elaborado pelo autor

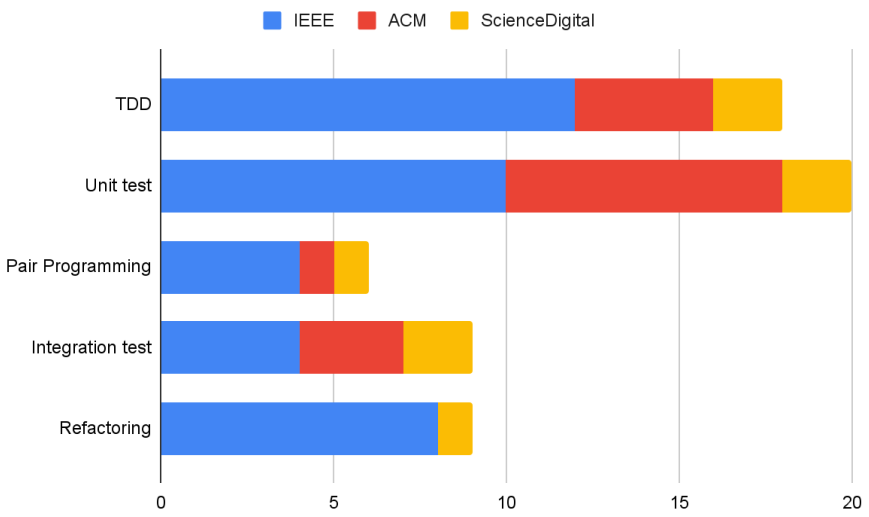
Os terceiro e quarto quadrantes que são relacionados a criticar o produto não tiveram tantas práticas citadas quando comparadas com os quadrantes com foco no suporte ao time.

3.7.1.1 Técnicas do Primeiro Quadrante de Testes Ágeis

Neste quadrante, onde o objetivo é o suporte ao time de desenvolvimento com foco na tecnologia, foi possível observar que as práticas de testes unitários, seguido de *Test-Driven Development (TDD)* foram as mais citadas nos estudos mapeados referente aos testes ágeis. Outras práticas como refatoração, testes de integração e programação em par também mostraram ser relevantes, como pode ser visto na Figura 8. Dos 32 estudos coletados para análise, cerca de 78% destes possuía pelo menos uma prática relativa ao primeiro quadrante de testes ágeis.

No trabalho de Lazzarini Lemos et al. (2009), TDD é uma técnica de desenvolvimento ágil muito popular, onde o desenvolvedor escreve os casos de testes antes de desenvolver o código. Em relação aos testes unitários, segundo Fischbach et al. (2020), estes são geralmente utilizados e implementados por desenvolvedores para manter a qualidade do código, testando por unidades.

Figura 8: Quantidade de práticas dentro do quadrante 1

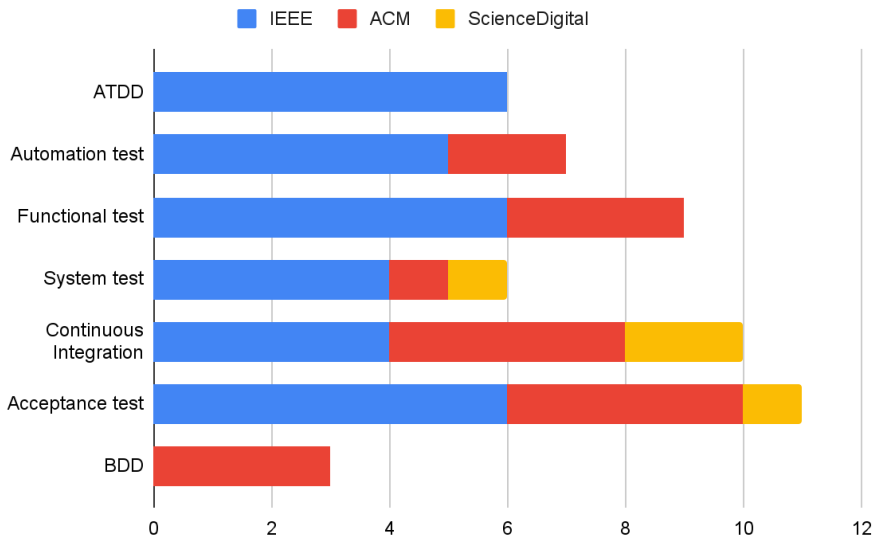


Fonte: Elaborado pelo autor

3.7.1.2 Técnicas do Segundo Quadrante de Testes Ágeis

Para o segundo quadrante, um total de 23 trabalhos apresentaram práticas de testes com o objetivo de suporte ao time e com o foco no negócio, sendo os testes de aceitação, prática de integração contínua e os testes funcionais as práticas mais relevantes referentes a esse quadrante, presentes em 10 e 8 artigos, respectivamente. A prática de testes automáticos também esteve presente em grande parte destes estudos, como pode ser visto na Figura 9.

Figura 9: Quantidade de práticas dentro do quadrante 2

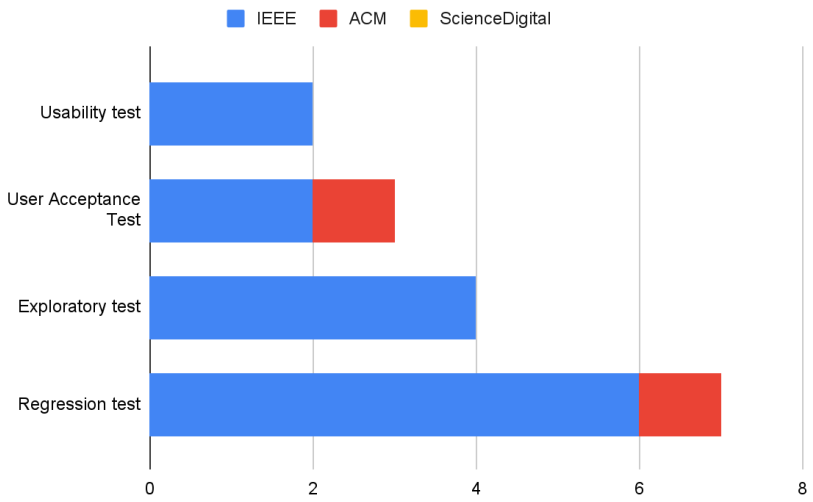


Fonte: Elaborado pelo autor

3.7.1.3 Técnicas do Terceiro Quadrante de Testes Ágeis

Seguindo para os testes que tem como objetivo criticar o produto mas ainda com o foco nos negócios, os testes de regressão e exploratórios foram referenciados em 7 e 4, respectivamente, sendo as técnicas mais relevantes dos estudos mapeados para o terceiro quadrante, como mostra a Figura 10. Outras práticas de testes ágeis também foram encontradas, como o testes de usabilidade, testes de aceitação do usuário e o teste de transição de estado. No total foram selecionados 12 estudos que contemplavam tais técnicas.

Figura 10: Quantidade de práticas dentro do quadrante 3



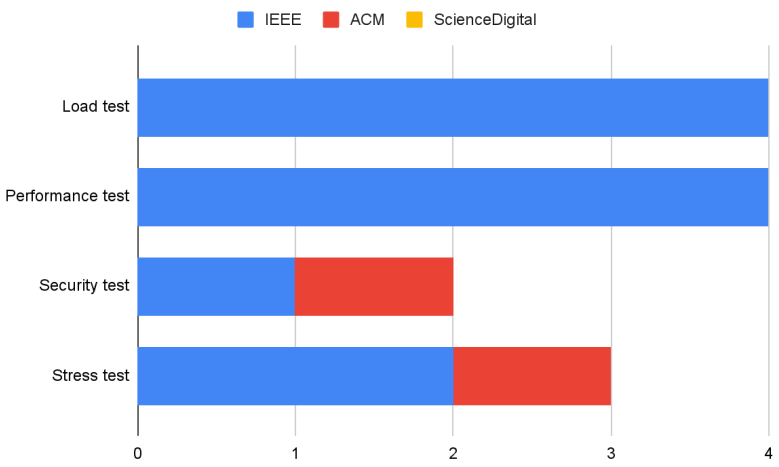
Fonte: Elaborado pelo autor

Em Ngah et al. (2017), o teste de regressão é descrito como uma validação para garantir que novos erros não sejam adicionados após mudanças de códigos em determinadas partes do software. Este tipo de teste permite a confiança e segurança em alterações de código, previsibilidade no lançamento do software, avaliação objetiva e redução de gastos em manutenção (SALOMÃO, 2016). Ainda para Salomão (2016), no âmbito de desenvolvimento de um software, é necessário a adoção de testes de regressão para que o time de testes sempre consiga validar o sistemas após modificações e ter um controle de qualidade para a entrega do produto.

3.7.1.4 Técnicas do Quarto Quadrante de Testes Ágeis

Para o último quadrante, foram encontrados 7 estudos que abordaram pelo menos uma das técnicas apresentadas na Figura 11. Estas técnicas têm como propósito focar na tecnologia para criticar o produto. Foram apresentados 5 tipos de testes diferentes, sendo os testes de carga e performance mais relevantes encontrados na pesquisa.

Figura 11: Quantidade de práticas dentro do quadrante 4



Fonte: Elaborado pelo autor

Para Hoffmann et al. (2014) a diferença entre os testes de carga e de performance se dá porque o teste de carga tem como foco a escalabilidade do sistema e o teste de performance na análise da quantidade tempo de execução das funcionalidades, sendo ambos os mais cruciais para sistemas embarcados em tempo real, mas que são geralmente ignorados ou despriorizados.

3.8 Considerações Finais

Neste capítulo foram apresentados os resultados do mapeamento sistemático da literatura referente às práticas de testes ágeis mais relevantes desde a introdução do modelo dos Quadrantes de Testes Ágeis. Com isso, ao todo foram selecionados e analisados 32 trabalhos para identificar essas técnicas e assim relacioná-las com cada quadrante do modelo.

O capítulo a seguir apresenta uma pesquisa em campo com profissionais de um projeto ágil no Centro de Estudos e Sistemas Avançados do Recife (CESAR), com o objetivo de identificar as aplicações das práticas de testes ágeis encontradas neste capítulo e assim fazer um diagnóstico com relação ao modelo de quadrantes de testes ágeis.

4. DIAGNÓSTICO DE TESTES ÁGEIS EM UM PROJETO CESAR

4.1 Introdução

Este capítulo apresenta os resultados de uma pesquisa realizada em campo com profissionais da indústria de software que fazem parte de um projeto ágil do Centro de Estudos e Sistemas Avançados do Recife (CESAR). Foi realizada uma entrevista qualitativa semi-estruturada para a coleta de dados.

O objetivo principal da pesquisa é entender a prática de testes ágeis dentro do processo de testes do projeto. Para isso, é importante entender quais testes são aplicados, a sua frequência dentro do processo do time e a sua importância na qualidade do produto.

4.2 Metodologia de Pesquisa

Entende-se que pesquisas são divididas em três grupos, sendo essas: exploratórias, explicativas e descritivas. A classificação da pesquisa tem como utilidade facilitar uma aproximação conceitual. Para esse trabalho, foi usado de uma pesquisa descritiva já que apenas uma amostra de população foi estudada, onde seu objetivo é de obter uma determinada população ou fenômeno o estabelecimento da descrição de suas características, de acordo com Gil (2008).

Dentro dessa pesquisa de campo foi empregada uma entrevista que seguiu duas técnicas de análise de dados: quantitativa e qualitativa. O uso da técnica qualitativa se deu para as perguntas subjetivas, onde o entrevistado poderia usar uma definição ou resposta particular para a pergunta. Já na análise quantitativa, foi usada para as perguntas objetivas.

4.3 Entrevista

A pesquisa teve como foco profissionais de desenvolvimento de software de um projeto que utiliza metodologia ágil dentro do Centro de Estudos e Sistemas Avançados do Recife. O projeto tem como escopo o desenvolvimento de produtos de software para um cliente parceiro do CESAR, tendo o comprometimento com a qualidade do que é entregue. O time é composto por 4 engenheiros de software, 3 engenheiros de testes, 3 designers e 1 gerente de projetos. Para esse trabalho, teve-se como objetivo entender, apenas, a relação entre

os desenvolvedores e dos testados com as práticas ágeis, já que são os principais executores de testes, de acordo com Duda (2017). Sendo assim, o trabalho contemplou um total de 7 entrevistas qualitativas.

Cada entrevista consistiu de 35 perguntas do tipo subjetivas e objetivas, sendo dividida em 3 partes. A primeira parte estava relacionada ao entendimento do perfil profissional, a segunda ao perfil do projeto e a última parte às práticas de testes ágeis, sendo dividida em 4 seções de acordo com cada quadrante pertencente ao modelo de Quadrantes de Testes Ágeis.

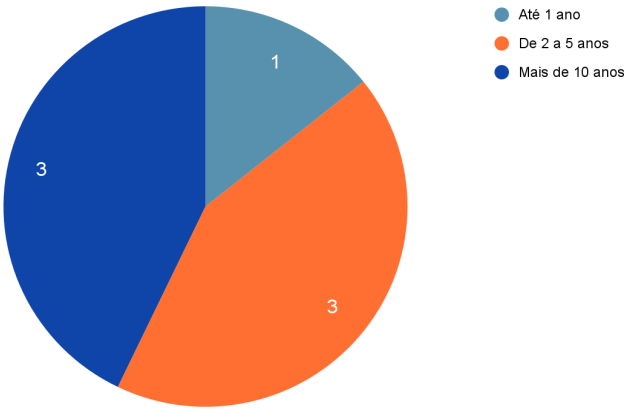
4.4 Amostra

Nesta seção serão apresentados os resultados obtidos da entrevista realizada e de acordo com as respostas dos entrevistados.

4.4.1 Experiências em Projetos de Software

A primeira fase da entrevista teve como foco o perfil profissional do entrevistado. O objetivo principal foi entender melhor o perfil de cada entrevistado, tendo como base a experiência de trabalho, o conhecimento e contato com metodologias ágeis. As duas primeiras perguntas foram mais diretas e buscaram saber o tempo de experiência com projetos de software e em projetos que utilizam metodologias ágeis. A Figura 12 mostra como estão distribuídos os entrevistados em relação ao tempo de experiência na área de desenvolvimento de softwares.

Figura 12: Experiência com projetos de software

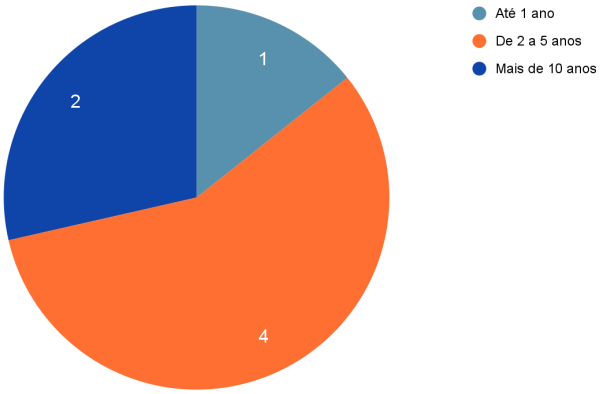


Fonte: Elaborado pelo autor

Pode-se notar que mais da metade do time é formado por profissionais com mais de 10 anos de experiência, seguido por profissionais com experiência de 2 a 5 anos, tendo apenas um profissional com 1 ano de experiência.

Em relação a experiência em projetos de software que utilizam alguma metodologia ágil, nota-se que o grupo estudado possui uma certa experiência, já que a proporção das respostas se mostrou bem semelhante com a experiência com projetos de software, diferenciando apenas que um dos entrevistados com mais de 10 anos de trabalho com projetos de software teve contato tem apenas de 2 a 5 anos de experiência com projetos ágeis, como é possível ver na Figura 13.

Figura 13: Experiência com projetos de software



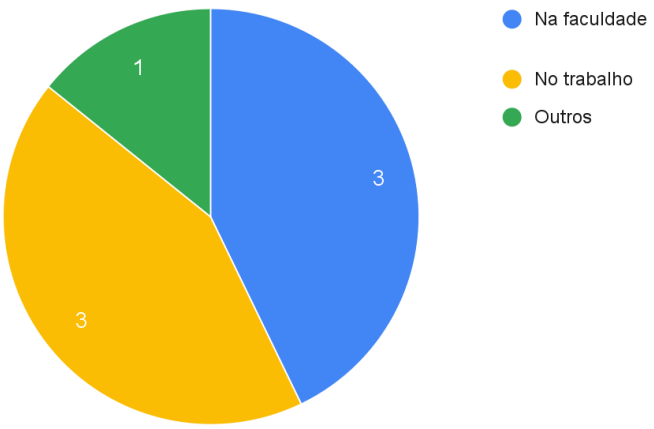
Fonte: Elaborado pelo autor

4.4.2 Experiências com Metodologias Ágeis

As próximas perguntas tiveram como objetivo entender o conhecimento do entrevistado com relação às metodologias ágeis. O entrevistado pôde responder mais de uma pergunta por não serem perguntas com apenas uma resposta possível.

De início, foi questionado onde o entrevistado conheceu a metodologia ágil de forma técnica para aplicá-la no seu cotidiano de trabalho. A Figura 14 mostra que grande parte aprendeu diretamente no âmbito de trabalho, sendo a palavra "treinamento" e "workshops" citadas em grande quantidade quando questionado como foi fornecido esse aprendizado. Em seguida, o aprendizado das metodologias ágeis se deu ainda na universidade durante projetos de Engenharia de Software. Por último, apenas um dos entrevistados mencionou que aprendeu de outra forma, sendo a internet o meio que foi utilizado para aprender e utilizar no trabalho.

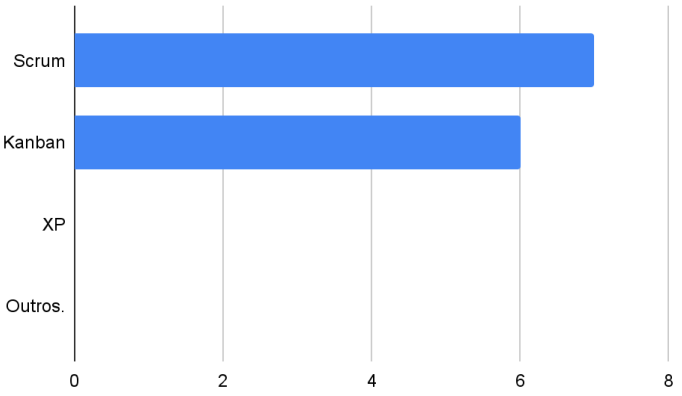
Figura 14: Onde aprendeu sobre metodologias ágeis



Fonte: Elaborado pelo autor

Em seguida, foi perguntado quais metodologias ágeis o entrevistado tinha um conhecimento técnico sobre. Para isso, foram listadas algumas metodologias mais utilizadas como Kanban, Scrum e XP somente como orientação de resposta, podendo ser adicionado alguma outra resposta, caso o entrevistado viesse a se lembrar.

Figura 15: Metodologias ágeis que já utilizou em projetos



Fonte: Elaborado pelo autor

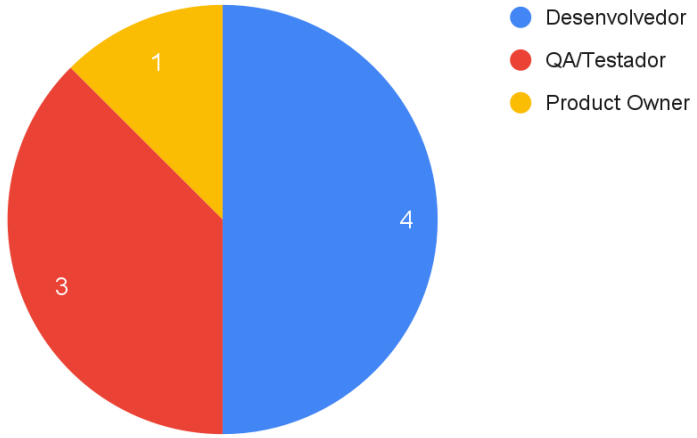
De acordo com a Figura 15, Scrum e Kanban foram as únicas citadas como utilizadas em projetos de software. Ao serem questionados, alguns dos entrevistados comentaram que conhecem razoavelmente sobre a metodologia XP e suas práticas, porém nunca estiveram em um projeto baseado nessa metodologia.

4.5 Entendimento do Perfil do Time

Essa fase da entrevista teve 2 perguntas principais para compreender o alinhamento do time em relação à metodologia utilizada no processo de desenvolvimento do projeto e também o papel de cada entrevistado no time. A primeira pergunta listou as mesmas metodologias elencadas na pergunta sobre o contato com metodologias ágeis. De acordo com 6 dos 7 entrevistados, o projeto não utiliza uma única metodologia e sim uma junção de práticas do Scrum com práticas apresentadas pela metodologia ágil Kanban. O outro entrevistado comentou que Scrum seria a metodologia do projeto, mas que algumas práticas estavam relacionadas a outra metodologia, porém não saberia informar qual especificamente.

A segunda pergunta teve como objetivo mapear os papéis de cada entrevistado para facilitar o relacionamento de cada papel com as práticas de testes ágeis a serem perguntadas na Seção 4.6. Na Seção 4.3 foi feito um breve resumo da divisão de papéis do time, podendo ser vista também na Figura 16.

Figura 16: Papéis dos entrevistados no time estudado



Fonte: Elaborado pelo autor

Devido à sua maior experiência, um dos entrevistados acumula os papéis de desenvolvedor e Product Owner. No papel de QA, um dos entrevistados é responsável por organizar e definir os processos de testes no projeto.

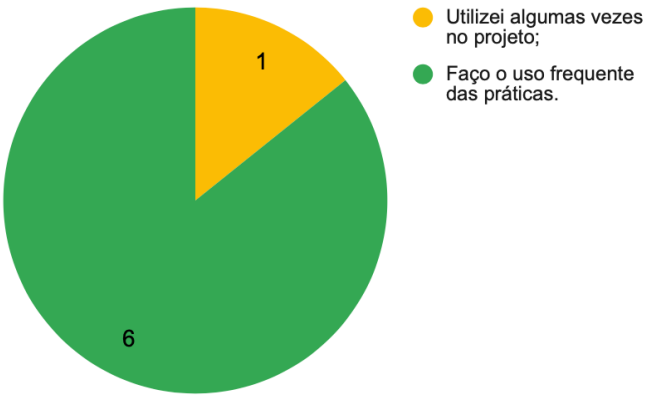
4.6 Práticas de Testes Ágeis

A última etapa da entrevista foi composta de 29 questões com o objetivo de identificar as técnicas de testes ágeis de acordo com o modelo dos Quadrantes de Testes Ágeis (QTA). De início, foi feito um resumo sobre os testes ágeis e sobre o QTA a fim de contextualizar os entrevistados sobre tais temas para ajudar em suas respostas. Essa etapa da entrevista foi dividida em 4 sub etapas de acordo com as práticas relacionadas a cada quadrante separadamente, sendo levantadas as seguintes questões por quadrante na entrevista:

- Quais técnicas de testes ágeis você conhece?
- Quais técnicas de testes ágeis são utilizadas no seu projeto?
- Qual a frequência da utilização dessas práticas?
- Qual o papel responsável por essas práticas?
- Qual o motivo da utilização de tais práticas?
- Qual o motivo da não utilização de tais práticas?
- Qual a importância da utilização dessas práticas no projeto?

Como pergunta introdutória foi questionada a relação de cada entrevistado com práticas de testes ágeis para entender se o time estava bem alinhado com o uso de pelo menos alguma técnica. Na Figura 17 é possível observar que apenas um dos questionados disse fazer o uso em momentos específicos no projeto, se o restante tem contato frequente com práticas de testes ágeis. Isso demonstra que o time é formado por profissionais que possuem uma familiaridade com o tema a ser questionado.

Figura 17: Contato dos entrevistados com testes ágeis

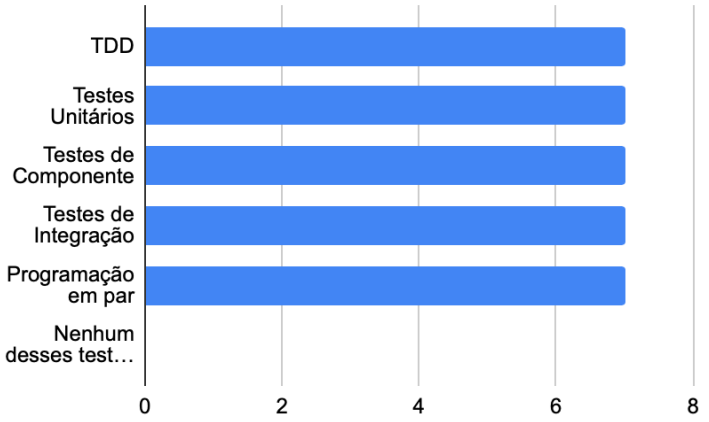


Fonte: Elaborado pelo autor

4.6.1 Práticas Relativas ao Primeiro Quadrante de Testes Ágeis

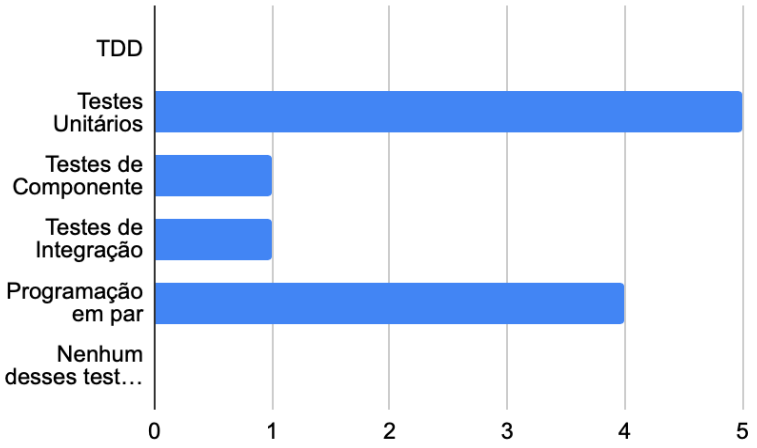
A Figura 18 mostra o conhecimento do time em relação às técnicas relativas ao Primeiro Quadrante de Testes Ágeis, onde todos os entrevistados responderam que conhecem, pelo menos na teoria, o que seria cada uma das práticas apresentadas. Quando questionados sobre a utilização das práticas no cenário atual do projeto, os testes unitários e a programação em par foram as mais citadas, como visto na Figura 19.

Figura 18: Distribuição do conhecimento do time com relação às técnicas relativas ao Primeira Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

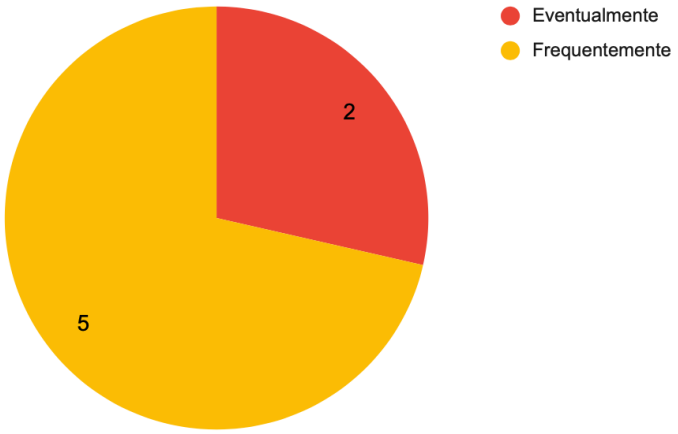
Figura 19: Distribuição da prática do time com as técnicas relativas ao Primeira Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

Os testes de componente e de integração foram citados apenas uma vez, não ficando claro se essa técnica é realmente usada no projeto ou se foi apenas um equívoco do entrevistado.

Figura 20: Frequência de utilização das técnicas relativas ao Primeiro Quadrante de Testes Ágeis

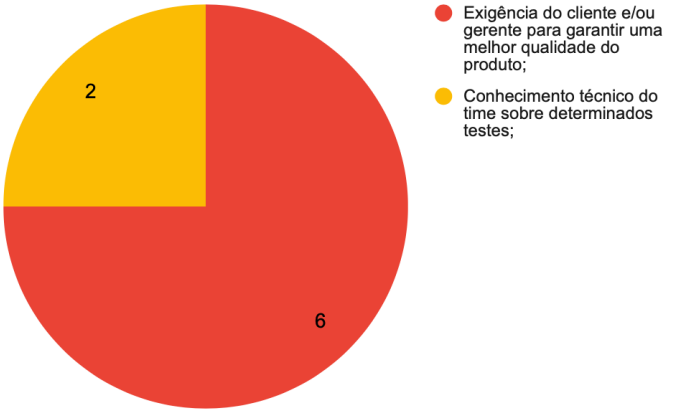


Fonte: Elaborado pelo autor

Sobre a frequência da utilização de tais técnicas relativas ao Primeiro Quadrante de Testes Ágeis, cinco dos sete entrevistados confirmaram que fazem o uso frequente das práticas no projeto e o restante disse que o uso é eventual. Para os entrevistados que responderam "Eventualmente" foi indagado o motivo da não frequência do uso. O motivo seria que em algumas ocasiões, era necessário priorizar outras atividades ao invés da escrita de novos testes.

Na Figura 21, a exigência do cliente foi a principal razão do uso desses testes, sendo essa resposta relacionada com os testes unitários por todos os entrevistados. O conhecimento técnico do time também se mostrou uma motivação relevante para o uso de tais práticas.

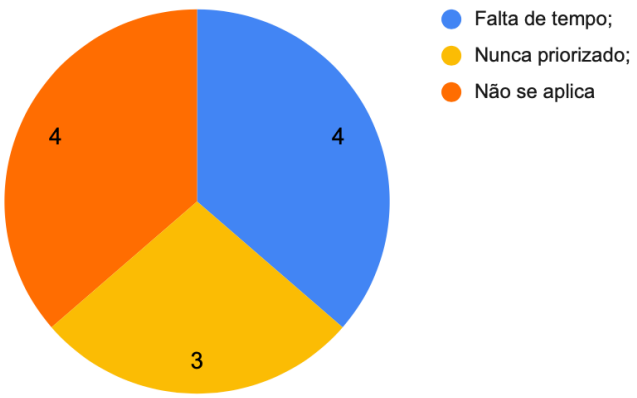
Figura 21: Razão da utilização das técnicas relativas ao Primeiro Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

Quando questionados sobre a não utilização de outras práticas de testes, quatro respostas estavam relacionadas com a falta de tempo para a aplicação delas devido à quantidade de features a serem desenvolvidas. O TDD, por exemplo, se encaixa perfeitamente nesse cenário, já que seria custoso mantê-lo dado o problema de tempo. O restante dos entrevistados responderam que não possuíam resposta para esse quadrante, como visto na Figura 22.

Figura 22: Razão da não utilização das técnicas relativas ao Primeiro Quadrante de Testes Ágeis



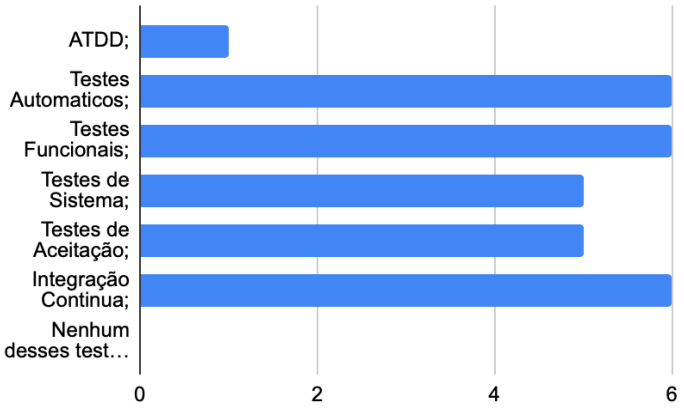
Fonte: Elaborado pelo autor

A última pergunta: "Qual a importância da utilização dessas práticas no projeto?" teve como característica dar mais liberdade e espaço ao entrevistado de dar sua visão, sem alternativas pré estabelecidas, com o objetivo de notar os principais pontos positivos quando os testes do Primeiro Quadrante são aplicados. Para alguns entrevistados "essencial" seria a palavra que mais se relaciona com esse quadrante, pois o básico da aplicação seria garantido e problemas poderiam ser evitados e encontrados já em fase de desenvolvimento. Outra resposta que se mostrou bem aplicada seria em relação à segurança do código, já que um código sem cobertura de testes está mais vulnerável a *bugs*.

4.6.2 Práticas Relativas ao Segundo Quadrante de Testes Ágeis

A Figura 23 mostra o conhecimento dos entrevistados sobre as técnicas relativas ao Segundo Quadrante de Testes Ágeis. Para esse quadrante os testes automáticos, funcionais e a técnica de integração contínua se mostraram as mais relevantes. Logo em seguida os testes de sistema e aceitação também foram bastante comentados. Para a prática do ATDD apenas um entrevistado disse já ter tido algum tipo de contato prévio.

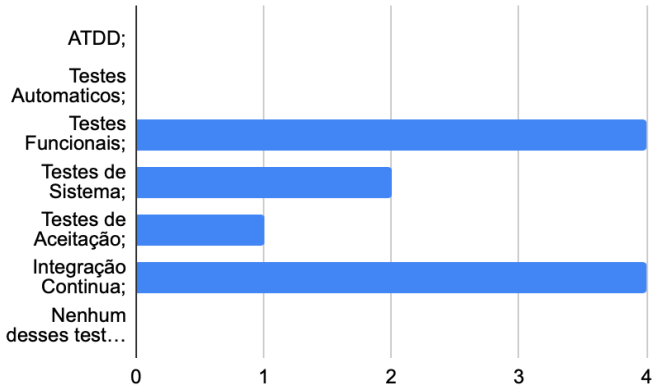
Figura 23: Contato dos entrevistados com testes ágeis



Fonte: Elaborado pelo autor

No contexto do projeto, os testes funcionais e a prática de integração contínua foram citados 4 vezes, de acordo com a Figura 24. Em contraste com ao encontrado na pergunta anterior, os testes automáticos ou automatizados não são utilizados pelo projeto.

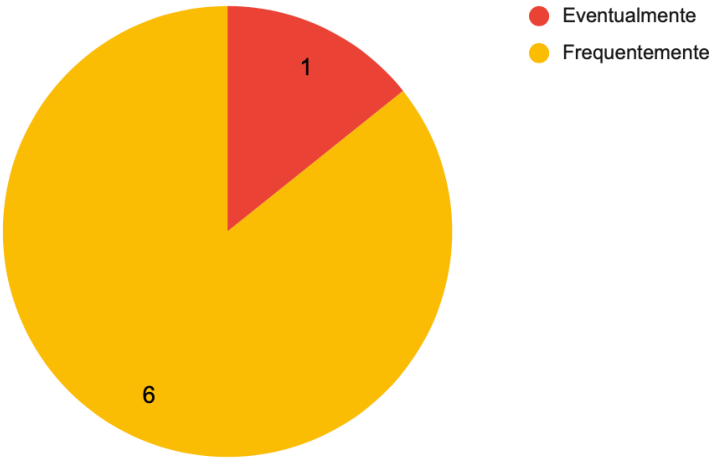
Figura 24: Contato dos entrevistados com testes ágeis



Fonte: Elaborado pelo autor

Em relação à frequência desses testes usados no processo de testes do projeto, seis entrevistados responderam que é frequente o uso destas práticas, como visto na Figura 25.

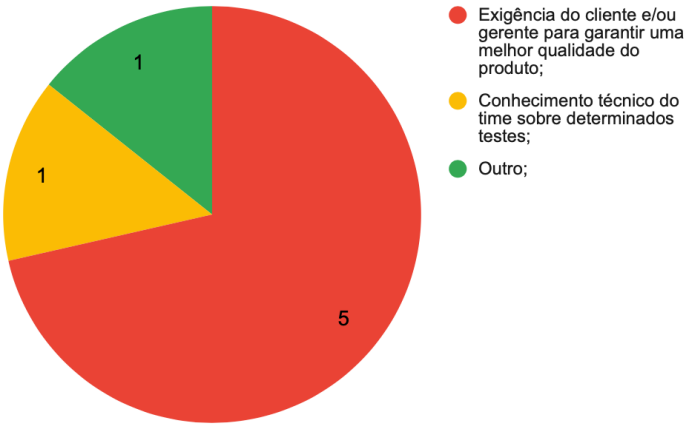
Figura 25: Contato dos entrevistados com testes ágeis



Fonte: Elaborado pelo autor

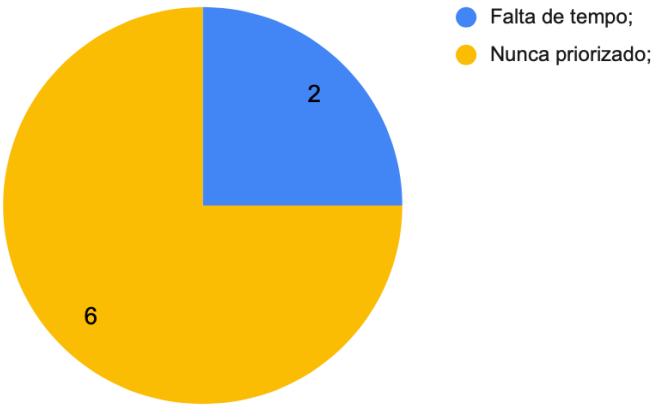
Como visto na Figura 26, o principal motivo da utilização dessas práticas seria a exigência do cliente com a qualidade do produto. Para a maior parte dos entrevistados, a não utilização das outras práticas como os testes automáticos não é considerada crítica para garantir a qualidade do produto. Outra razão seria a falta de tempo por ter um custo atrelado ao seu uso.

Figura 26: Razão da utilização das técnicas relativas ao Segundo Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

Figura 27: Razão da não utilização das técnicas relativas ao Segundo Quadrante de Testes Ágeis



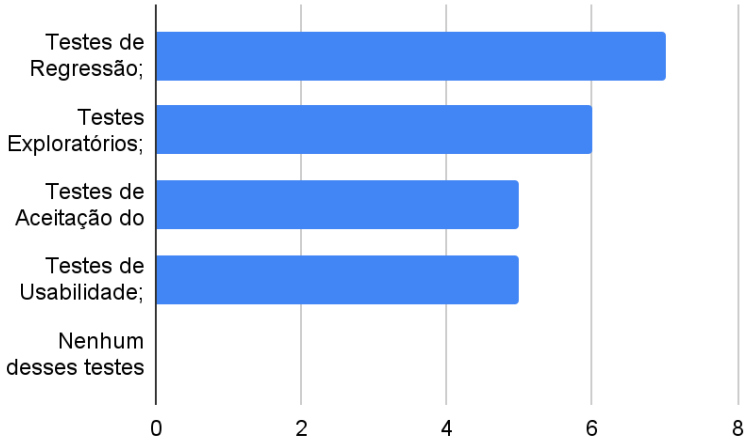
Fonte: Elaborado pelo autor

Sobre a importância que as práticas desse quadrante trazem para o projeto, os entrevistados comentaram que os testes que garantam o comportamento correto das funcionalidades são essenciais para manter o nível de qualidade. Para alguns entrevistados, o uso de integração contínua é um importante ponto positivo adotado pelo projeto para integração do código durante o processo de build. De acordo com o sentimento da maioria dos entrevistados, a automação de testes iria ajudar a reduzir o esforço dos testadores para cada versão lançada e assim permitir a execução de mais testes sem necessidade de ocupar um QA para re-testar funcionalidades base da aplicação.

4.6.3 Práticas Relativas ao Terceiro Quadrante de Testes Ágeis

Na Figura 28 mostra que, todos os testes associados a esse quadrante se mostraram bem reconhecidos pelo grupo analisado, sendo o teste de regressão reconhecido por todos entrevistados. Para os testes exploratórios, seis entrevistados comentaram que conheciam essa prática por ser amplamente usada em todos os projetos que tiveram contato. Para um dos entrevistados, a definição e a importância dos testes exploratórios não era clara. Os testes de aceitação e testes de usabilidade têm um total de cinco entrevistados que conhecem.

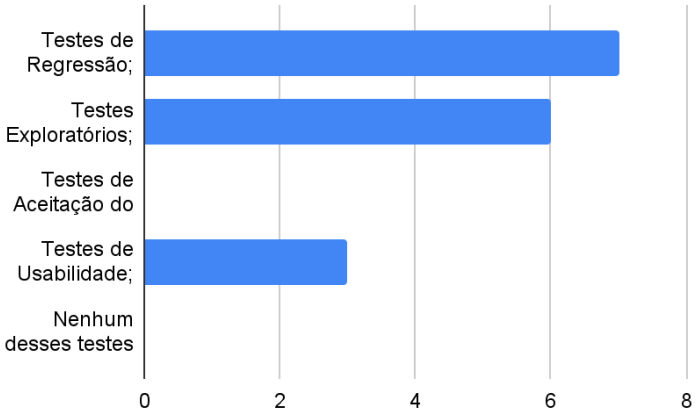
Figura 28: Distribuição do conhecimento do time com relação às técnicas relativas do Terceiro Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

No time estudado, os testes de regressão e testes exploratórios são vistos como fundamentais no processo de testes do projeto. Para esses testes foram obtidos todos os votos, exceto o exploratório, pois um entrevistado não tinha certeza da sua utilização no projeto. Para os testes de usabilidade, apenas três entrevistados acreditam na sua utilização dentro do projeto, de acordo com a Figura 29.

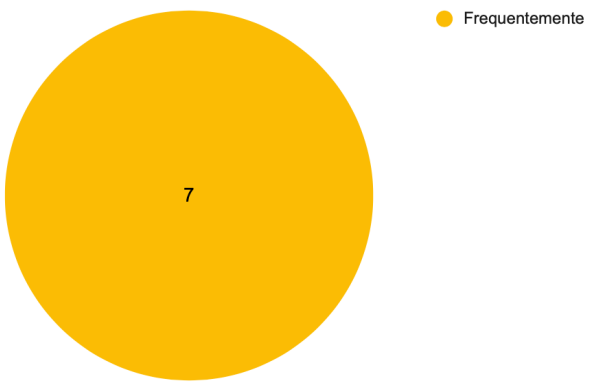
Figura 29: Distribuição da prática do time com as técnicas relativas ao Terceiro Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

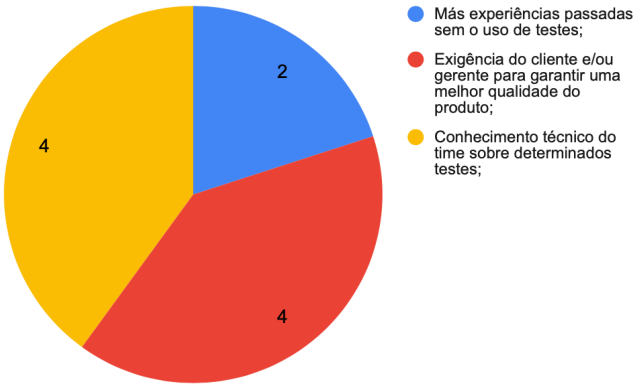
Conforme a Figura 30, para o time as práticas de teste relativas ao quadrante 3 são frequentemente executadas durante o processo de testes da aplicação. A razão de sua aplicação se mostrou diversa na Figura 31, 4 votos estão relacionados a uma exigência de qualidade do próprio cliente, já para os outros 4 entende-se que o conhecimento técnico do time foi o motivo principal para estruturar um processo de testes realizado pelos testadores do projeto e os outros 2 diz que más experiências passados podem ter ajudado a ter esse cuidado com a qualidade do produto. Na Figura 32 mostra que, para os testes de aceitação do usuário e usabilidade, seis entrevistados não souberam responder, já o restante apontou que não foi priorizado pelo cliente.

Figura 30: Frequência de utilização das técnicas relativas ao Terceiro Quadrante de Testes Ágeis



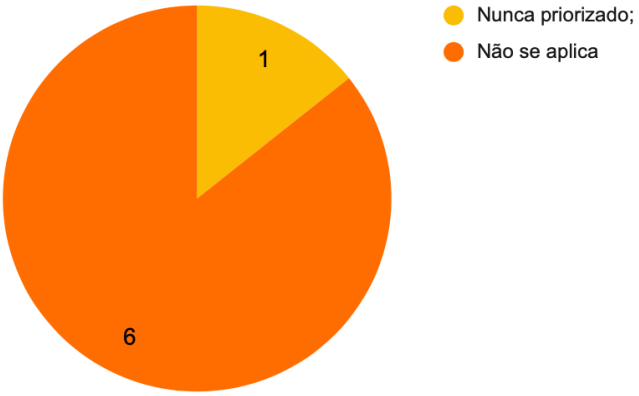
Fonte: Elaborado pelo autor

Figura 31: Razão da utilização das técnicas relativas ao Terceiro Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

Figura 32: Razão da não utilização das técnicas relativas ao Terceiro Quadrante de Testes Ágeis



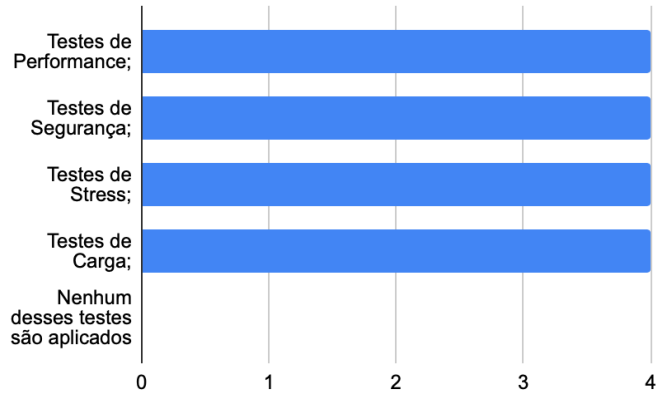
Fonte: Elaborado pelo autor

Como questão final desse quadrante, os entrevistados foram questionados sobre a importância de adotar tais testes no seu processo de qualidade. Para alguns tem um impacto muito positivo, pois permite que bugs que foram introduzidos no desenvolvimento de alguma feature ou algum comportamento diferente do planejado sejam capturados antes do usuário final ter acesso à versão desenvolvida. "Garantir que refatorações não afetem partes do sistema" foi um outro ponto levantado, já que constantemente o time de desenvolvimento está aprimorando seu código e ter essa garantia pelos testes de regressão acaba cobrindo esse cenário.

4.6.4 Práticas Relativas ao Quarto Quadrante de Testes Ágeis

O quarto quadrante se mostrou o menos relevante dentro do projeto. Na Figura 33 pode ser visto que todas as práticas levantadas na entrevista se mostraram conhecidas, onde cada entrevistado tinha conhecimento teórico ou prático de pelo menos uma das técnicas.

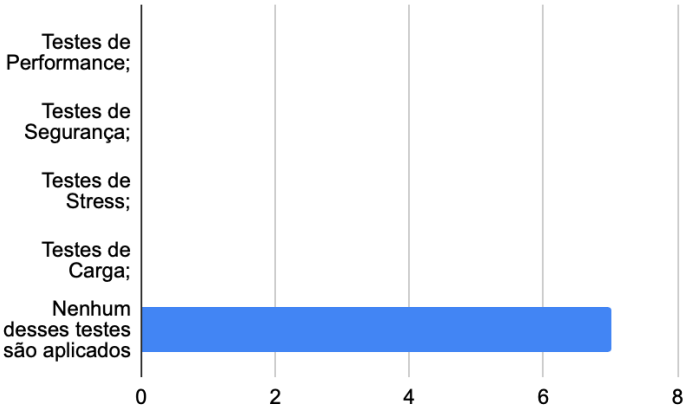
Figura 33: Distribuição do conhecimento do time com as técnicas relativas ao Quarto Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

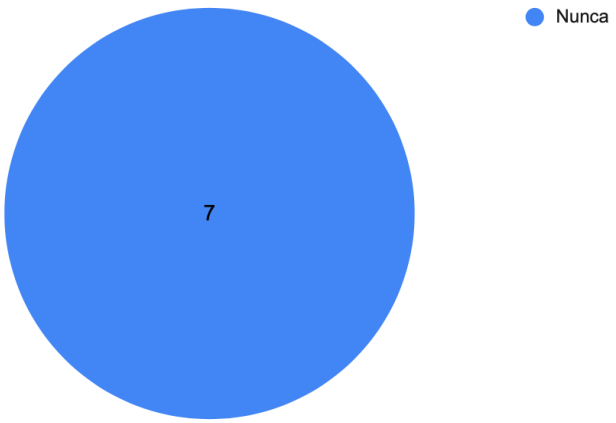
Em relação ao uso dessas práticas ou alguma outra que fosse relativa ao Quarto Quadrante de Testes Ágeis, nenhuma se mostrou aplicada pelo grupo estudado, como pode ser visto nas Figuras 34, 35 e 36.

Figura 34: Distribuição da prática do time com as técnicas relativas ao Quarto Quadrante de Testes Ágeis



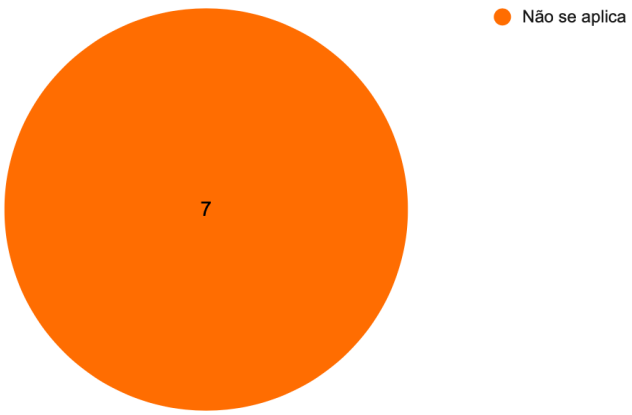
Fonte: Elaborado pelo autor

Figura 35: Frequência de utilização das técnicas relativas ao Quarto Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

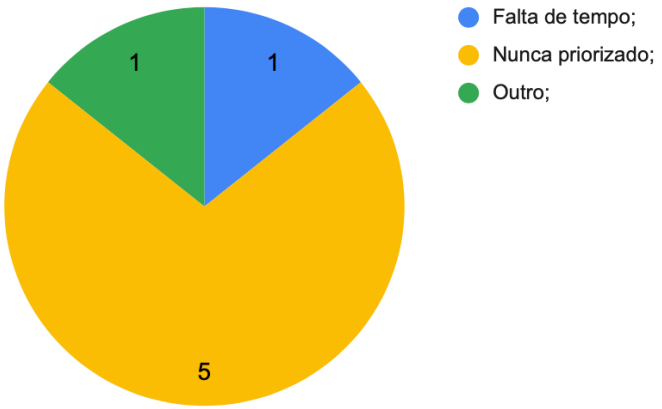
Figura 36: Razão da utilização das técnicas relativas ao Quarto Quadrante de Testes Ágeis



Fonte: Elaborado pelo autor

Na Figura 37 são mostradas as razões para as práticas relativas a esse quadrante não serem aplicadas dentro do projeto, sendo a não priorização o principal motivo para 5 das respostas. Os outros entrevistados responderam que o motivo da não utilização seria falta de tempo para a aplicação desses testes. Outra razão seria devido o projeto estar em uma fase inicial, não mostrando tanto impacto a curto prazo.

Figura 37: Razão da não utilização das técnicas relativas ao Quarto Quadrante de Testes Ágeis



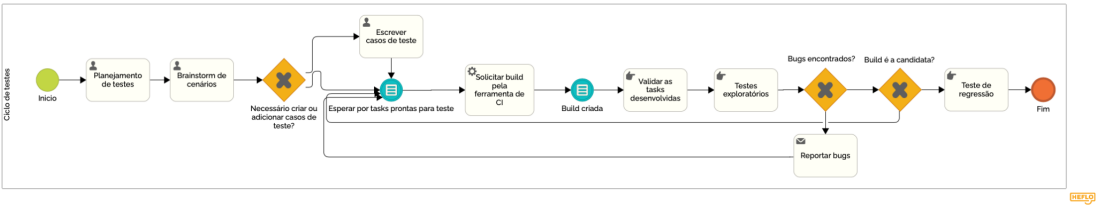
Fonte: Elaborado pelo autor

Mesmo com a não utilização de tais práticas de testes, foi comentado pelos entrevistados que seria de grande importância a sua aplicação devido aos problemas de performance que o produto possui. Um outro problema visto pelos entrevistados seria a não visibilidade de possíveis problemas relacionados ao suporte de uma grande quantidade de carga, já que por estar em fase de desenvolvimento não se tem grande uso da aplicação por usuários. Também foi relatado que o sistema já demonstra possuir alguns problemas iniciais em relação à sua performance, onde em determinados fluxos é possível encontrar gargalos e lentidão, mas que hoje em dia não são detectados por testes direcionados a esses problemas e sim por testes exploratórios realizados.

4.7 Processo de Teste AS-IS

Para melhor compreensão do atual processo de testes do projeto, foi realizada uma modelagem AS-IS por meio da criação de um BPMN a partir do que foi fornecido pelo time (Figura 38).

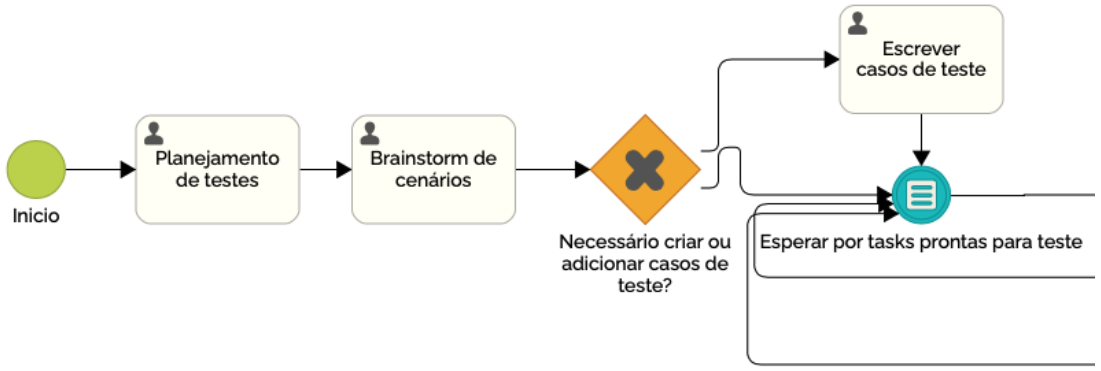
Figura 38: BPMN do processo de testes do projeto



Fonte: Elaborado pelo autor

No início de cada sprint, o time de qualidade faz o plano de teste referente às novas funcionalidades a serem desenvolvidas. Após isso, é executado um brainstorming para entender quais cenários que as mudanças afetam e quais novos fluxos podem ser incrementados, assim é definido se é preciso criar ou atualizar os casos de testes da suite, caso seja necessário os testadores iniciam a atividade de escrita, como demonstrado na Figura 39.

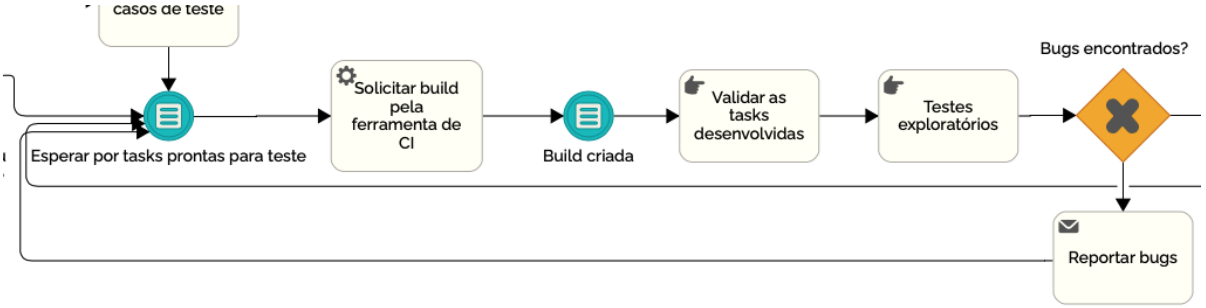
Figura 39: BPMN da fase inicial do processo de testes



Fonte: Elaborado pelo autor

Durante todo esse processo de desenvolvimento, o time de teste fica encarregado de aguardar a finalização de tarefas que sejam testáveis, tais como a implementação de uma tela nova, quando é então solicitada a geração de uma *build* pela ferramenta de Integração Contínua usada, como mostrado na Figura 40. Após uma versão nova do sistema ser criada, são executados os testes planejados para essas mudanças, bem como os exploratórios para encontrar cenários não mapeados e para garantir que os fluxos principais não foram afetados. Caso bugs sejam encontrados o time usa uma ferramenta para reportar o problema.

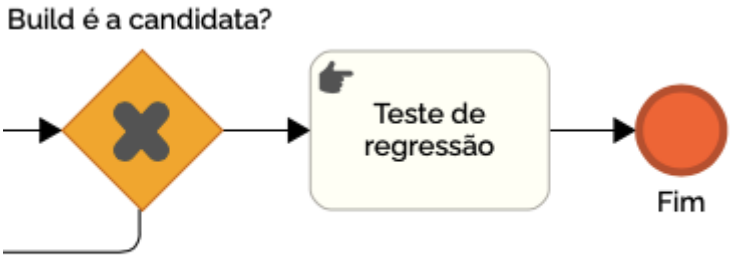
Figura 40: BPMN da fase de validação do processo de testes



Fonte: Elaborado pelo autor

Na Figura 41 é possível ver que ao final da validação dessa versão, é verificado se a mesma é candidata a ser lançada, com base na criticidade dos bugs que foram e nas atividades de desenvolvimento a serem implementadas para a sprint atual. No momento que se tem uma versão candidata para lançamento, é executada a suíte de regressão para assim garantir que defeitos não conhecidos não estejam presentes na aplicação.

Figura 41: BPMN da fase final do processo de testes



Fonte: Elaborado pelo autor com base

4.8 Considerações Finais

Com a entrevista realizada, conclui-se que, em geral, o projeto do CESAR analisado se mostra alinhado com aplicação dos testes ágeis para garantir a qualidade interna e externa do produto. O time demonstrou ter experiência referente ao uso de práticas de ágeis, o que pode ter facilitado na adoção de algumas práticas já na fase inicial do projeto. Além disso, os três primeiros quadrantes se mostraram presentes durante o ciclo de desenvolvimento do produto, sendo levantados alguns problemas, pelos entrevistados, no desenvolvimento de novos testes unitários. Algumas outras práticas como TDD, referente ao primeiro quadrante, e testes automáticos, do segundo quadrante, são práticas que não estão presentes atualmente no projeto, mas que parte dos entrevistados declararam ser importantes para sua adoção no futuro.

Para testes do quarto quadrante, nenhum se mostrou aplicado por o projeto estar em uma fase inicial. Contudo, de acordo com Lucena e Tizzei (2016), mesmo a falta do seu uso ser bem comum no mercado brasileiro, a implantação em fases finais de desenvolvimento pode gerar prejuízos. Isso se dá por ser necessário realizar mudanças referentes a essa área de

execução. Essa problemática foi um dos pontos comentados pelo Product Owner do projeto, que em projetos passados, teve más experiências pela negligência do projeto em testes não funcionais.

O próximo capítulo apresentará um conjunto de melhorias propostas, de acordo com o estado da arte, para que o time consiga utilizá-las de modo a agregar valor ao seu processo de qualidade.

5. PROPOSTA DE MELHORIA

5.1 Introdução

O diagnóstico realizado inicialmente serviu para o entender o estado-da-prática do processo de teste deste projeto ágil com base no modelo de Quadrantes de Testes Ágeis, para que fosse possível visualizar as falhas e pontos de melhorias, que são importantes para o processo de produção das melhorias de acordo com as práticas encontradas no estado-da-arte.

As melhorias propostas neste trabalho estão presentes em cada quadrante, pois nota-se a importância de projetos ágeis aplicarem testes para garantir a sua qualidade. Como exemplo, pode-se ver o caso aplicado ao quarto quadrante, onde foi possível observar uma despriorização de testes deste quadrante.

5.2 Pontos de Melhoria

A seguir serão apresentados os principais pontos de melhoria e recomendações de práticas de testes relativas a cada seguimento dos Quadrantes de Testes Ágeis.

5.2.1 Estudar Viabilidade do uso do TDD

Em concordância com a entrevista realizada, compreende-se que o time de desenvolvimento demonstrou ter uma base para a aplicação do TDD, dado que todos os entrevistados responderam que possuem conhecimento acerca dessa prática. De acordo com um dos desenvolvedores entrevistados, em um projeto anterior ao que se encontra atualmente, o desenvolvimento guiado a testes era uma de suas principais práticas de teste ágil aplicada.

Com isso, percebeu-se que havia um grande impacto positivo para o desenvolvimento da aplicação, gerando diversos benefícios em relação à qualidade com que o produto era desenvolvido. A existência de um membro do time com experiência prática pode facilitar uma possível adesão a essa prática, sendo facilitador de implementação e de colaboração com seus companheiros de equipe de desenvolvimento. De acordo com Duda (2017), na análise da utilização de práticas do Quadrante 1 dos Quadrantes de Testes Ágeis no Porto Digital, onde está localizado o C.E.S.A.R, é relatado que apenas 25% dos entrevistados aplicam essa prática em seus projetos, mostrando que o seu uso não é corriqueiro.

Em conformidade com Khanam (2017), são elencados os ganhos de aplicar essa prática em um projeto de software, tais esses como:

- O aprimoramento na qualidade do software.
- A qualidade do código.
- Maior facilidade para refatoração do código.
- O custo para fazer a manutenção do código é diminuido.
- O time possui mais confiança com o que é desenvolvido.
- O custo é minimizado tendo em vista que o re-trabalho e ajustes de bugs são reduzidos.

Porém, ainda segundo Khanam (2017), abordar essa técnica no processo de desenvolvimento de um software também pode gerar impactos negativos, tais como:

- Em projetos com uma frequência maior de mudança de design ou com uma grande mudança futura dos requisitos, será necessário um gasto maior de tempo para atualizações dos testes;
- É importante salientar que o desenvolvimento guiado a testes deve consumir pelo menos 15% tempo extra para a escrita desses testes, sendo importante diminuir a quantidade de funcionalidade entregues por iteração;
- A falta de desenvolvedores com uma expertise nessa prática pode ser um risco;
- A produtividade do time de desenvolvimento tende a ser afetada negativamente.

Com base no diagnóstico desenvolvido, algumas das limitações podem ser minimizadas, já que parte do time de desenvolvimento possui uma experiência com a prática, podendo assim gerenciar melhor essas limitações. Outrossim, é fundamental o entendimento da prioridade estabelecida pelo time e cliente, se a diminuição de escopo de novas funcionalidades deve ser um problema para aumentar a qualidade do código desenvolvido.

A partir destas conclusões, dos benefícios e limitações especificados, é recomendado o estudo da viabilidade de adesão do TDD no projeto. Dessa forma, buscando aprimorar a qualidade do produto e a diminuição de bugs encontrados de forma tardia, evitando-os antes do processo de *building*.

5.2.2 Manter Frequência no Desenvolvimento de Testes Unitários

A partir da análise do diagnóstico realizado, entende-se que o desenvolvimento de testes unitários é uma exigência do cliente do projeto, tendo em vista o padrão de qualidade estabelecido pelo mesmo. Na entrevista, o time de desenvolvimento relatou que possui o entendimento acerca dessa exigência, e com isso faz a aplicação de novos testes, visando garantir que o código seja testado.

No entanto, em relação à frequência do uso de testes unitários, relatado por alguns entrevistados uma certa preocupação com a diminuição da escrita de novos casos de testes de unidade com o passar das iterações. É possível constatar que a falta de tempo é um dos problemas principais para o time manter a frequência de criação de novos testes unitários, pois com o desenvolvimento de novas funcionalidades e correção de bugs encontrados pelo time de teste, a escrita tende a não ser priorizada.

Kua (2018) apresenta algumas razões para os testes unitários acabarem sendo negligenciados, com o passar do tempo, em projetos de software. Os dois principais motivos seriam:

- Os benefícios de seu uso acabam não sendo reconhecidos por completo;
- O tempo que é disponibilizado para o desenvolvimento de novos *unit tests* não é suficiente ou não é contabilizado.

A partir desse cenário, entende-se que a dificuldade relatada pelo grupo analisado está de acordo com o que é encontrado em Kua (2018). Assim, é recomendado que o time de desenvolvimento possa ter maior disponibilidade de tempo em prol da frequência de testes unitários. Durante o planejamento de atividades, a escrita de testes não deve ser negligenciada, evitando a ocorrência da priorização apenas da entrega de *features.*

5.2.3 Estudar a Viabilidade de Testes Automáticos

De acordo com as entrevistas realizadas, foi constatado que o time referenciou algumas desvantagens devido a falta de testes automatizados para a validação das funcionalidades do produto e para detecção de *bugs*. Ainda segundo as entrevistas, os testes automatizados de interface poderiam colaborar no atual processo de qualidade do software,

dado que seria possível executar uma suíte automatizada de testes com mais frequência, a fim de assegurar automaticamente que uma nova versão de *build* não afetou nenhuma outra funcionalidade, poupando esforço do time de QA para testes manuais repetitivos.

Conforme Crispin e Gregory (2009), algumas das motivações para automatizar testes são as execuções lentas e repetitivas dos testes manuais. A automação surge como solução para proporcionar resultados de testes mais rápidos e poder ser executado por qualquer pessoa. Essas razões estão extremamente alinhadas com o desejo do time. Com o crescimento do software e com novos casos de testes para serem criados a cada iteração, o tempo de execução de um ciclo de teste tende a aumentar, dessa forma fazendo com que seja necessário ter novos testadores para dividir tarefas. A repetitividade dos testes manuais faz com que a qualidade do software venha a diminuir devido a situações de se tornar uma atividade tediosa, gerando maior erros durante as validações e passando despercebidos de bugs menos visíveis.

A automação pode providenciar *feedbacks* mais rápidos para o time, já que alterações de código ou refatorações não devem afetar o resultado do teste. Essa detecção antecipada de bugs faz com que a qualidade da aplicação aumente e que o time de desenvolvimento possa trabalhar antecipando correções antes do *code-freeze.*

Visto que as motivações do time de automatizar testes de interface estão de acordo com as principais razões para implementar testes automáticos, é importante seguir recomendações básicas para criação de um processo de implementação. Em Collins e De Lucena (2012), é descrito um experimento de adoção de automação de testes em dois projetos ágeis, onde foram constatados algumas recomendações e lições aprendidas que podem ajudar no gerenciamento de riscos em projetos com o mesmo intuito, sendo essas:

- **Colaboração é um dos principais motivos de sucesso para a adoção de testes automatizados em um time ágil.** O envolvimento dos testadores em discussões de *backlog review*, design de interfaces, configuração de ambiente e na arquitetura do sistema, gera uma proximidade maior do time, evitando uma separação entre desenvolvedores e testadores. A programação em par, algo que já é aplicado pelos desenvolvedores do projeto, deve ser de grande importância na configuração do ambiente de automação, inspeção de código, revisão e codificação.

- **A escolha das ferramentas de teste com base na estratégia.** Para o autor, a escolha das ferramentas para automatizar testes é muito importante para seu sucesso, onde em projetos ágeis é recomendado o uso de ferramentas com uma curva de aprendizado menor

para minimizar os gastos com tempo na configuração do ambiente de testes. Isso se destaca, pois em um ambiente ágil as ferramentas devem ser flexíveis para se adequarem ao processo de testes do projeto.

- **A automação de testes não deve ser limitada apenas aos testadores.** É necessário que qualquer membro do time possa facilmente ter acesso a todo ambiente de automação, seja para rodar testes ou ver os resultados, já que definir essa atividade e concentrar o conhecimento em um membro ou dois pode gerar riscos em projetos ágeis.

- **Documentação e feedback de informação.** Segundo o autor, existem ferramentas que devem ser usadas para a documentação e gerenciamento de informações relacionadas aos testes. Ferramentas de controle de teste podem trazer diversos benefícios coletando informações como: *bug tracking*, falhas detectadas, especificação de casos de teste e outros. Além disso, é possível a geração de relatórios de acordo com a execução de uma suíte de testes já automatizada, agregando na qualidade ao produto.

Além das recomendações propostas nesse trabalho, é fundamental o estudo da viabilidade de implementação de um processo de automação, como também a alocação de esforços para o desenvolvimento destes testes.

5.2.4 Introdução dos Testes de Aceitação do Usuário

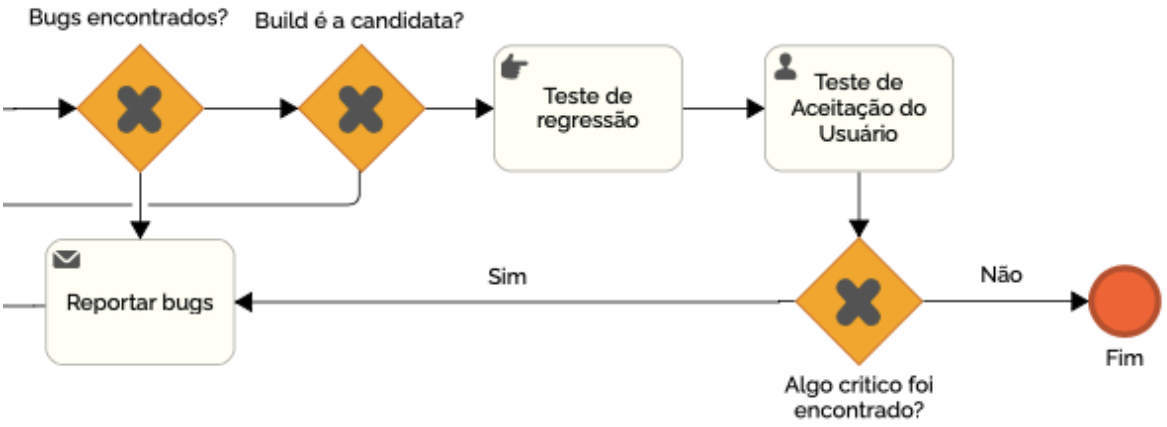
Conforme o diagnóstico realizado no Terceiro Quadrante de Testes Ágeis, percebe-se que o projeto demonstrou o uso de práticas de testes relativos a esse quadrante. Porém, de acordo com as entrevistas, compreende-se que atualmente o uso de testes de aceitação do usuário (UAT) não estão dentro do processo de teste.

Segundo Crispin e Gregory (2009), os testes UAT servem tanto para aplicações de grande porte como para aplicações de uso interno. Tendo em vista que, no final de cada nova versão do software, quem irá usufruir das funcionalidades são os usuários, é necessário que os próprios possam verificar se as funcionalidades do sistema estão de acordo, funcionando da forma esperada. Ainda de acordo com Crispin e Gregory (2009), existem diferentes recomendações para diferentes tipos de softwares: produtos internos e produtos com uma base grande de usuários. Para os softwares de uso interno é possível definir formalmente como deve ser realizada a execução de teste, definir qual grupo de usuários deve executar os testes e prestar suporte aos usuários em caso de dúvidas. Dessa forma, é possível ter uma relação dos resultados de cada teste executado, entendendo a criticidade de cada problema reportado e

sendo priorizado durante a fase de desenvolvimento. Outrossim, em produtos com uma grande base de usuários, é recomendado o uso de testes Alfa e Beta para conseguir coletar os *feedbacks* dos usuários reais.

Conforme o atual processo de teste do projeto, não existe uma fase de testes de aceitação do usuário. Vê-se a necessidade de que seja colocado em prática o uso desse teste antes da *release* oficial do sistema para os usuários. Após os testes de regressão realizados pelo time de QA em uma versão candidata, os usuários pré-definidos para testar poderiam receber essa versão e executar testes de acordo com suas expectativas, assim reportando os *feedbacks* e possíveis *bugs* escapados. Ao fim, o time conseguirá entender quais problemas afetam de forma crítica o usuário, priorizando a correção de tais problemas e ajustando os casos de testes para detectar cenários como esses. A representação dessa fase na modelagem do processo é vista na Figura 42.

Figura 42: Fase do teste UAT no final do processo



Fonte: Elaborado pelo autor

A partir da implementação de UAT, alguns benefícios podem ser percebidos pelo time por completo, pois esse teste irá prover a visão dos usuários finais diante a aplicação, o que pode-se gerar um contraste com a visão atual dos colaboradores do projeto, melhorias na usabilidade do software, evitar problemas em produção que afetem diretamente os usuários, entre outros pontos importantes.

5.2.5 Por em Prática os Testes Relativos ao Quarto Quadrante de Testes Ágeis

O projeto mostrou-se ausente de práticas de testes relativas ao quarto quadrante, visto que nenhum dos tipos de teste listados na entrevista é aplicado no processo de qualidade. Para Crispin e Gregory (2009), é bem comum que projetos considerem problemas não funcionais como de baixo risco para seu produto e com isso, durante o planejamento de testes, técnicas e práticas que atuem na detecção desses problemas sejam despriorizadas e descartadas. Portanto, ainda segundo o autor, é importante para garantir a qualidade do processo de teste que todos os quadrantes do modelo Quadrantes de Testes Ágeis sejam aplicados. Para isso, a aplicação de práticas de testes do quarto quadrante é fundamental para cumprir esse requisito.

Segundo a entrevista realizada, mais da metade do time tem conhecimento sobre os testes de performance, carga, stress e segurança, sendo um ponto positivo para o processo de melhoria, já que esses seriam os testes mais relevantes deste quadrante, segundo o mapeamento realizado pelo trabalho. Foi diagnosticado que existem dois problemas principais relacionados com a falta desses testes, sendo: problemas de performance do sistema em cenários que possuam uma grande quantidade de informações a serem mostradas, além da falta de visibilidade do time de qual seria o impacto proveniente do uso do sistema, por uma grande quantidade de usuários.

Ressalta-se que para tais problemas, é recomendado o uso de testes de performance, pois esse tipo de teste deve ajudar a identificar problemas de gargalo do sistema, assim conseguindo ajudar na tomada de decisões sobre os problemas encontrados. Para avaliar o comportamento do sistema em momentos de estresse, processamento de dados e o impacto da alta quantidade de usuários, é importante aplicar testes de carga, de acordo com (CRISPIN e GREGORY, 2009).

Para Sarojadevi (2011), existem diversos benefícios com a adoção de testes de performance e carga, sendo estes alguns dos principais citados:

- Melhorar a confiabilidade do produto já que *deadlocks* podem ser evitados. O gerenciamento do tempo de resposta do programa, e o suporte a escalabilidade do produto são pontos cobertos por esses testes.

- Garantir maior segurança do software, pois testes que suportam a performance tendem a detectar *overflows* de memória e assim evitar que vulnerabilidades sejam expostas aos usuários.

- É possível monitorar e medir diversos problemas referentes ao desempenho do software como inconsistência de dados, violação de ordem de bytes e problemas com memória e assim consequentemente controlá-los.

5.3 Considerações Finais

As propostas elencadas tiveram como objetivo principal atender os problemas e dificuldades relatados pelos entrevistados diante das práticas relativas a cada quadrante dos Quadrantes de Testes Ágeis.

Ressalta-se que, um futuro estudo de viabilidade, executado pelo time, aponta a necessidade de uma avaliação entre as partes interessadas e assim gerenciar os possíveis riscos do projeto em prol da qualidade. A implementação de testes relativos ao quarto quadrante são as mais prioritárias para o projeto no cenário atual. A razão dessa criticidade é devido a ausência completa de testes que suportem essa área.

6. CONCLUSÃO

6.1 Considerações Finais

Esse trabalho buscou fazer um paralelo entre o estado da prática e o estado da arte, identificando as principais dificuldades encontradas e propor melhorias para o processo de teste do projeto. Para isto, foi executado um mapeamento sistemático da literatura e, também, uma pesquisa em campo no formato de entrevista, com profissionais que atuam em um projeto ágil dentro da instituição CESAR, localizada no Porto Digital do Recife.

O modelo de Quadrantes de Testes Ágeis foi importante para identificar as principais práticas de testes ágeis, e assim entender a sua importância em um processo de testes. No primeiro quadrante é possível encontrar práticas como TDD e testes unitários ajudando na especificação de requisitos para o time de desenvolvimento. Os testes de aceitação do usuário e usabilidade que suportam a qualidade de um produto em relação às necessidades do usuário. Além do quarto quadrante, que critica o produto com base em problemas não funcionais, sendo eles de desempenho ou segurança.

A pesquisa realizada identificou quais as práticas de testes utilizadas no projeto, e como eram aplicadas. Porém, pode ser visto que, algumas práticas importantes para o processo de teste que não eram aplicadas mesmo o grupo tendo conhecimento técnico acerca. A partir do diagnóstico realizado com os resultados obtidos comparando com o encontrado na literatura, foi possível detectar possíveis pontos de melhoria do processo de qualidade do software, como: a recomendação do uso de práticas e ajustes no atual processo de testes.

6.2 Limitações

Houve algumas limitações relacionadas ao mapeamento realizado no Capítulo 3. Teste de software é um tema muito relevante na Engenharia de Software. Com isso, foi necessário reduzir a quantidade de repositórios para apenas 3. Essa limitação pode ter deixado de fora alguns trabalhos que seriam relevantes para a análise e composição deste trabalho.

Além disso, era desejável fazer uma avaliação de viabilidade das propostas de melhorias e entender qual seria o impacto real em seu projeto. Contudo a falta de tempo não permitiu tal aplicação. Também, nota-se que, os resultados obtidos com esse diagnóstico não podem ser

generalizados para outros projetos da mesma instituição ou para empresas do Porto Digital do Recife.

6.3 Trabalhos Futuros

Como trabalho futuro é recomendada a realização de um estudo de viabilidade para implementação das melhorias propostas. A partir disso, deve-se avaliar o impacto na qualidade do processo de testes com essa implementação.

Referências

ISTQB - International Software Testing Qualifications Board. **Certified Tester Syllabus Foundation Level CTFL 3.1**. Disponível em: <https://bstqb.org.br/b9/doc/syllabus_ctfl_3.1br.pdf>. Acesso em 10 out. 2022.

GIL, Antônio Carlos. **Como Elaborar Projetos de pesquisa**. 2002. Disponível em: <https://professores.faccat.br/moodle/pluginfile.php/13410/mod_resource/content/1/como_ela borar_projeto_de_pesquisa_-_antonio_carlos_gil.pdf>. Acesso em: 10 out. 2022.

DIGITAL.AI. **15th Annual State of Agile Report**, 2021. Disponível em: <https://info.digital.ai/rs/981-LQX-968/images/RE-SA-15th-Annual-State-Of-Agile-Report.pdf>. Acesso em: 05 out. 2022.

WATKINS, John. **Agile testing: how to succeed in an extreme testing environment.** Cambridge University Press, 2009.

AGILE MANIFESTO. **Manifesto for Agile Software Development.** Disponível em: <http://agilemanifesto.org/>. Acesso em: 10 out. 2022.

CRISPIN, Lisa; GREGORY, Janet. **Agile testing: A practical guide for testers and agile teams**. Pearson Education, 2009.

BRERETON, Pearl et al. **Lessons from applying the systematic literature review process within the software engineering domain**. Journal of systems and software, v. 80, n. 4, p. 571-583, 2007.

FOWLER, Martin; HIGHSMITH, Jim. **The agile manifesto**. Software Development, v. 9, n. 8, p. 28-35, 2001.

MOLYNEAUX, Ian. **The art of application performance testing: Help for programmers and quality assurance.** O'Reilly Media, Inc. 2009.

WHITMAN, Michael E.; MATTORD, Herbert J. **Principles of information security.** Cengage Learning, 2011.

KITCHENHAM, Barbara; CHARTERS, Stuart. **Guidelines for performing systematic literature reviews in software engineering**. Keele University, UK, v. 9, 2007.

LUCENA, Percival; TIZZEI, Leonardo. **Applying Software Craftsmanship Practices to a Scrum Project: an Experience Report**. In: Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES 2016), 73-84.

MELO et al. **The evolution of agile software development in Brazil**. Journal of the Brazilian Computer Society, v. 19, n. 4, p. 523-552, 2013. Disponível em: https://doi.org/10.1007/s13173-013-0114-x . Acesso em: 10 out. 2022.

LEITE, Renato Atouguia. **SaaS Process: Um Processo de Desenvolvimento para Software Como Serviço**. 2022. Dissertação de Mestrado. Universidade Federal de Pernambuco. Disponível em: https://repositorio.ufpe.br/handle/123456789/45774. Acesso em: 10 out. 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª ed. São Paulo: Pearson Education Brasil, 2011. 529 p.

WAKODE, Rajat B.; RAUT, Laukik P.; TALMALE, Pravin. **Overview on kanban methodology and its implementation**. IJSRD-International Journal for Scientific Research & Development, v. 3, n. 02, p. 2321-0613, 2015.

ALAIDAROS, Hamzah; OMAR, Mazni; ROMLI, Rohaida. **The state of the art of agile kanban method: challenges and opportunities**. Independent Journal of Management & Production, v. 12, n. 8, p. 2535-2550, 2021.

BECK, Kent. **Extreme programming explained: embrace change**. addison-wesley professional, 2000.

RUNESON, Per. **A survey of unit testing practices**. IEEE software, v. 23, n. 4, p. 22-29, 2006.

LIMA, Vagner Carlos Marcolino. **Programação em par: investigando sua eficácia perante tarefas de modelagem e construção de software**. 2013. Dissertação de Mestrado. Universidade Tecnológica Federal do Paraná.

HOFFMANN, Luiz Felipe Simoes et al. **Applying acceptance test driven development to a problem based learning academic real-time system**. In: 2014 11th International Conference on Information Technology: New Generations. IEEE, 2014. p. 3-8.

CASTRO, Ana Karina Silva de. **Testes Exploratórios: Características, Problemas e Soluções**. 2018. Trabalho de Conclusão de Curso. Universidade Federal do Rio Grande do Norte.

BACH, James. **Exploratory testing explained** . 2003 Disponível em: < http://www.satisfice.info/articles/et-article.pdf > Acessado em: 10 out. 2022

FELDERER, Michael et al. **Security testing: A survey.** In: Advances in Computers. Elsevier, 2016. p. 1-51.

DUDA, JOÃO GUILHERME FARIAS. **Agile Testing: Uma Análise da Prática no Porto Digital do Recife**. 2017. Trabalho de Graduação de Curso. Universidade Federal de Pernambuco.

KHANAM, Zeba; AHSAN, Mohammed Najeeb. **Evaluating the effectiveness of test driven development: advantages and pitfalls**. International Journal of Applied Engineering Research, v. 12, n. 18, p. 7705-7716, 2017.

LAZZARINI LEMOS, Otávio Augusto et al. **Applying test-driven code search to the reuse of auxiliary functionality**. In: Proceedings of the 2009 ACM symposium on Applied Computing. 2009. p. 476-482.

FISCHBACH, Jannik et al. **What Makes Agile Test Artifacts Useful? An Activity-Based Quality Model from a Practitioners' Perspective**. In: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). 2020. p. 1-10.

NGAH, Amir; MUNRO, Malcolm; ABDALLAH, Mohammad. **An overview of regression testing**. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), v. 9, n. 3-5, p. 45-49, 2017.

SALOMÃO, Renata Gonzaga. **Análise da relevância de testes de regressão para o mercado de desenvolvimento de software do Triângulo Mineiro**. 2016.

KUA, Patrick. **Unit Testing**. Disponível em: https://www.thekua.com/publications/AppsUnitTesting. pdf, 1995.

SAROJADEVI, H. **Performance testing: methodologies and tools**. Journal of Information Engineering and Applications, v. 1, n. 5, p. 5-13, 2011.

SWAMINATHAN, Vasu. ATDD – **An applied approach to continuous delivery**. Disponível em: https://www.aspiresys.com/articles/atdd-applied-approach-to-continuous-delivery.pdf Acessado em: 10 out. 2022

FERREIRA, Kátia Gomes; DE CURSO, Monografia de Final; DA SILVA, Clarindo Isaías Pereira. **Teste de usabilidade**. Monografia de Final de Curso: Especialização em Informática, Universidade de Minas Gerais. Belo Horizonte, Brasil, 2002.

NIELSEN, Jakob. **Usability engineering**. Morgan Kaufmann, 1994.

COLLINS, Eliane Figueiredo; DE LUCENA, Vicente Ferreira. Software test automation practices in agile development environment: An industry experience report. In: 2012 7th International Workshop on Automation of Software Test (AST). IEEE, 2012. p. 57-63.

APÊNDICES

APÊNDICE A — ROTEIRO DA ENTREVISTA

Disponivel em:

https://docs.google.com/presentation/d/13x3QIU5Zt9BsZN8luag0upqU8TYpbJuVfLAdQIN6dO8/edit?usp=sharing

Apêndice B — Estudos primários selecionados

IDENTIFICAÇÃO	ANO	TÍTULO DO ARTIGO	ACESSO
A01	2010	Empirical Studies on Quality in Agile Practices: A Systematic Literature Review	https://ieeexplore.ieee.org/document/5654783
A02	2011	Common Agile Practices in Software Processes	https://ieeexplore.ieee.org/document/6092587
A03	2011	Testing in a Agile Product Development Environment: An Industry Experience Report	https://ieeexplore.ieee.org/document/5985897
A04	2012	Agile Testing: Past, Present, and Future -- Charting a Systematic Map of Testing in Agile Software Development	https://ieeexplore.ieee.org/document/6298092
A05	2012	Development of Auxiliary Functions: Should you be agile? An Empirical Assessment of Pair Programming and Test-first Programming	https://ieeexplore.ieee.org/document/6298092
A06	2012	Automated Acceptance Testing as an Agile Requirements Engineering Practice	https://ieeexplore.ieee.org/document/6149535
A07	2012	Influences on Agile Practice Tailoring in Enterprise Software Development	https://ieeexplore.ieee.org/document/6170020
A08	2013	Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences	https://ieeexplore.ieee.org/document/6612876
A09	2013	A test automation framework in POCT system using TDD techniques	https://ieeexplore.ieee.org/document/6612876
A10	2014	Applying Acceptance Test Driven Development to a Problem Based Learning Academic Real-Time System	https://ieeexplore.ieee.org/document/6822167
A11	2015	Practices to Make Agile Test Teams Effective: Challenges and Solutions	https://ieeexplore.ieee.org/document/7227526
A12	2016	Applying Agile Practices To Avoid Chaos In User Acceptance Testing: A case Study	https://ieeexplore.ieee.org/document/7480122
A13	2016	A Preliminary Analysis of Various Testing Techniques in Agile Development - A Systematic Literature Review	https://ieeexplore.ieee.org/document/7783283

A14	2017	Software Testing: The State of the Practice	https://ieeexplore.ieee.org/document/8048665
A15	2018	Design and Application on Agile Software Exploratory Testing Model	https://ieeexplore.ieee.org/document/8048665
A16	2018	Software testing: Survey of the industry practices	https://ieeexplore.ieee.org/document/8048665
A17	2021	A Study on Assets Applied in Exploratory Test Design And Execution: an Interview Application	https://ieeexplore.ieee.org/document/9637359
A18	2022	Automated Unit Testing and Test-Driven Development Approach to Teaching C++	https://ieeexplore.ieee.org/document/9796750
A19	2020	What Makes Agile Test Artifacts Useful?: An Activity-Based Quality Model from a Practitioners' Perspective	https://dl.acm.org/doi/10.1145/3382494.3421462
A20	2016	Towards Agile Testing for Railway Safety-critical Software	https://dl.acm.org/doi/10.1145/3382494.3421462
A21	2018	Assessing Agile Testing Practices for Enterprise Systems: A Survey Approach	https://dl.acm.org/doi/10.1145/3382494.3421462
A22	2012	Software Test Automation Practices in Agile Development Environment: An Industry Experience Report	https://dl.acm.org/doi/10.5555/2663608.2663620
A23	2011	A literature review of agile practices and their effects in scientific software development	https://dl.acm.org/doi/10.5555/2663608.2663620
A24	2012	Empirical studies of embedded software development using agile methods: a systematic review	https://dl.acm.org/doi/10.5555/2663608.2663620
A25	2014	Legend: an agile DSL toolset for web acceptance testing	https://dl.acm.org/doi/10.5555/2663608.2663620
A26	2013	Industrial study on test driven development: challenges and experience	https://dl.acm.org/doi/10.5555/2662528.2662535
A27	2021	Barriers to Shift-Left Security: The Unique Pain Points of Writing Automated Tests Involving Security Controls	https://dl.acm.org/doi/10.1145/3475716.3475786
A28	2021	Application of agile methods in traditional logistics companies and logistics startups: Results from a German Delphi Study	https://www.sciencedirect.com/science/article/abs/pii/S0164121221000479

A29	2018	Optimization of Software Testing	https://www.sciencedirect.com/science/article/pii/S1877050918308743
A30	2022	Software Testing from an Agile and Traditional view	https://www.sciencedirect.com/science/article/pii/S1877050922007219
A31	2020	Are unit and integration test definitions still valid for modern Java projects? An empirical study on open-source projects	https://www.sciencedirect.com/science/article/abs/pii/S0164121219301955

Apêndice C — Técnicas relativas aos estudos primários selecionados

TÉCNICA	QUADRANTE	ESTUDOS PRIMÁRIOS
TDD	Q1	A01, A02, A03, A04, A05, A07, A08, A09, A18, A21, A24, A26, A28, A29
Testes Unitários	Q1	A03, A04, A08, A10, A13, A14, A16, A18, A19, A20, A21, A22, A23, A24, A26, A27, A29, A31
Programação em Par	Q1	A01, A02, A05, A07, A23, A28
Testes de Integração	Q1	A03, A13, A14, A16, A20, A21, A27, A29
Refatoração	Q1	A01, A02, A08, A09, A28
ATDD	Q2	A04, A06, A08, A10, A15
Automação de Testes	Q2	A08, A11, A13, A16, A18, A21, A22,
Testes Funcionais	Q2	A03, A04, A13, A14, A15, A21, A22, A25
Testes de Sistema	Q2	A13, A14, A16, A20, A29
Integração Contínua	Q2	A04, A07, A20, A22, A27, A28, A30
Testes de Aceitação	Q2	A04, A08, A10, A14, A15, A19, A21, A22, A23, A29
BDD	Q2	A21, A25, A27
Testes de Usabilidade	Q3	A03, A10
Testes de Aceitação do Usuário	Q3	A03, A12, A20
Testes Exploratórios	Q3	A03, A13, A15, A17
Testes de Regressão	Q3	A03, A08, A11, A13, A14, A15,
Testes de Carga	Q4	A10, A13
Testes de Desempenho	Q4	A03, A04, A14, A16
Testes de Segurança	Q4	A16, A23
Testes de Stress	Q4	A03, A14, A23