AGILE TESTING E SUA IMPORTÂNCIA NO DESENVOLVIMENTO DE SOFTWARE

Aline Resende Neves[1], Marco Antônio Pereira Araújo[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – Câmpus Juiz de Fora

[2]Núcleo de Informática - Instituto Federal de Educação, Ciência e Tecnologia do Sudeste de Minas Gerais – Câmpus Juiz de Fora

alineneves.a21@gmail.com, marco.araujo@ifsudestemg.edu.br

Abstract. *This work was developed in order to draw a parallel in how testing activities are being redefined and reorganized in the agile context of software development. In addition to showing the importance of this movement to sustain and allow the increments not to lose quality due to the reduction of time (agility in sprints).*

Resumo. *Este trabalho foi desenvolvido visando traçar um paralelo em como as atividades de testes estão sendo redefinidas e reorganizadas no contexto ágil de desenvolvimento de software. Além de mostrar qual é a importância desse movimento para sustentar e permitir que os incrementos não percam qualidade em virtude da redução do tempo (agilidade nas sprints).*

1. Introdução

O *agile testing* é um processo iterativo e auto-organizado, em que a experiência e a colaboração governam a capacidade de entregar os itens planejados com eficácia, pegando a metodologia ágil e combinando-a com as melhores práticas dentro de um fluxo de trabalho para fornecer o máximo de feedback possível, ao mesmo tempo acessando o maior conhecimento e compreensão sobre a tecnologia no estado em que se encontra (GREGORY; CRISPIN, 2014).

Ao invés de esperar até o fim, imagine como seria atender ao processo de desenvolvimento com integração inovadora e planejada de testes ao longo do caminho, não deixando mais questões sobre qualidade em segundo plano, incluindo verificações de qualidade controladas em cada estágio em que as decisões estão sendo tomadas (MOHANTY; MOHANTY; BALAKRISHNAN, 2017).

Trabalhar desta forma tem a capacidade de produzir um produto utilizável com a máxima qualidade, um medidor de acerto e completude, dotado de conversas em planejamento de agilidade na entrega. Com essa filosofia, é possível responder a mais perguntas sobre qualidade na hora de lançar um software para o mundo e é sobre isso que esse trabalho irá tratar (MOHANTY; MOHANTY; BALAKRISHNAN, 2017).

1.1.Problema

Em um time de uma empresa específica havia muitas reaberturas de itens, demora na análise por um analista de qualidade de software (QA, do inglês *quality assurance*) e um distanciamento entre o QA e os desenvolvedores (dev, do inglês *developers*), com isso foi notado que poderia haver uma melhora nessa estrutura como um todo, para que houvesse um menor tempo de análise do QA e uma maior proximidade maior entre os QA's e devs, o que levaria há uma queda na reabertura de itens.

1.2.Justificativa

Pelo fato da autora deste trabalho atuar na equipe citada no tópico anterior este artigo é redigido para demonstrar a necessidade e importância de trazer o QA para mais perto das decisões dos itens de criação ou correção de software, para assim diminuir o tempo de análise, melhorar a qualidade de entrega e aproximar o QA do dev.

1.3.Objetivos

Os objetivos desse trabalho são demonstrar a diferença dos testes no modelo cascata e no *agile testing*, o primeiro pode ser trabalhado durante o desenvolvimento de um *software* e implantar uma parte do conceito do segundo em um time de uma empresa do setor de tecnologia da informação.

1.4.Hipótese

Para que o problema fosse solucionado, foi pensado em utilizar a metodologia ágil, que iria trazer o QA mais à frente das decisões dos itens, podendo melhorar o tempo de sua análise, a quantidade de reabertura dos itens, além de aproximar os QA's dos devs.

1.5.Estrutura do trabalho

Este trabalho está estruturado em capítulos, tendo no Capítulo 1 a introdução, no Capítulo 2 os fundamentos teóricos e conceitos iniciais, fomentando a base necessária para compreensão do mesmo, iniciando com uma apresentação sobre o conceito do modelo ágil, manifesto ágil, modelo cascata, agile testing e um comparativo entre o método ágil e o modelo cascata.

No Capítulo 3 descreve-se a metodologia utilizada, caracterizando a implantação da cultura ágil no time de uma empresa do setor de tecnologia da informação, e qual era o modelo utilizado. Em seguida, no Capítulo 4 são apresentados os resultados obtidos após a implantação, para cada etapa descrita, e no Capítulo 5 estão dispostas as conclusões do trabalho, englobando as vantagens e limitações do *agile testing*, e sugestões para futuras ações, seguido, por fim, das referências bibliográficas.

2. Revisão Sistemática da Literatura

Durante a contextualização do trabalho foi efetuada uma revisão sistemática da literatura em busca de técnicas e modelos que pudessem auxiliar na execução do mesmo. A tabela 1 apresenta o protocolo da revisão sistemática.

Tabela 1. Protocolo da Revisão Sistemática

Critério	Descrição
Seleção de fontes	Foi fundamentada a partir de itens que tratam da temática Agile Testing no desenvolvimento de software e qual a sua importância.
Palavras-chave	Agile Testing Practices, Importance, Quality, Software Development.
Idiomas dos Estudos	Português e Inglês.
Métodos de busca de fontes	As fontes foram acessadas via web. Foi utilizada a busca manual por livros já de conhecimento pessoal.
Listagem de fontes	Google Acadêmico.
Tipo dos Artigos/Livros	Teórico, estudo de casos.
Critérios de Inclusão e Exclusão de artigos	Os artigos estão disponíveis na web; os artigos consideram o estudo do Agile Testing no desenvolvimento de um software.

Fonte: (Autoria própria, 2022)

A pesquisa foi realizada através de buscas no Google Scholar utilizando a string: "agile testing practices" AND “quality” AND "software development" AND "importance", no qual resultou em 75 artigos. Após analisar todos os itens, foi constatado que apenas 8 seriam interessantes na utilização para a elaboração do artigo, sendo que desses 8, 2 são os livros das autoras Lisa Crispin e Janet Gregory, referências no assunto *agile testing*.

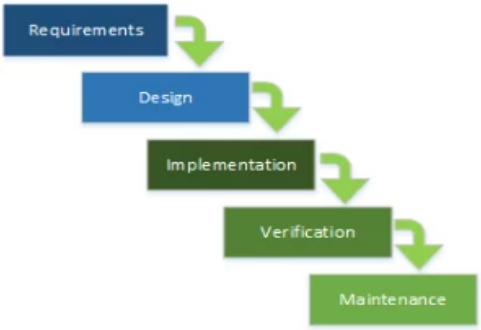
Estes itens foram escolhidos pelo fato de tratarem diretamente do *Agile Testing*, suas práticas e do porquê de sua importância, além de traçarem um paralelo entre a metodologia Ágil e o modelo Cascata. Com isso auxiliaram na escrita do artigo em questão.

2.1 Modelo Cascata (Waterfall)

O Modelo Cascata, ou Waterfall, foi o primeiro modelo de processo a ser introduzido e é muito simples de entender e usar. Em um modelo em cascata, cada fase deve ser concluída antes que a próxima fase possa começar e não haja sobreposição nas fases.

O modelo Waterfall é a primeira abordagem de Software Development Life Cicle (SDLC), em português Ciclo de Desenvolvimento de Software, e ilustra o processo de desenvolvimento de software em um fluxo sequencial linear. Isso significa que qualquer fase do processo de desenvolvimento começa apenas se a fase anterior estiver concluída. Neste modelo, as fases não se sobrepõem como mostra a Figura 1.

Figura 1 – Modelo cascata



Fonte: Adaptado de (DAVIES, 2018)

Como toda metodologia, o modelo cascata também possui fases, e nesse caso, são fases sequenciais. São elas:

1) Levantamento e análise de requisitos;
2) Projeto do Sistema (estudo das especificações de requisitos da primeira fase e elaboração o projeto do sistema);
3) Implementação (desenvolvimento e teste de cada unidade quanto à sua funcionalidade);
4) Integração e Testes;
5) Implantação do sistema;
6) Manutenção (correção de pequenos problemas e aprimoramento do produto);

2.2 Agile

Para entender um pouco mais sobre o modelo ágil (*agile*), é interessante entender primeiro o que é o SCRUM.

2.2.1 SCRUM

O SCRUM é uma estrutura para colaborações eficazes entre equipes que trabalham em produtos complexos. É um tipo de tecnologia ágil que consiste em reuniões, funções e ferramentas para ajudar as equipes a colaborar, estruturar e gerenciar melhor sua carga de trabalho. Embora seja usado com mais frequência por equipes de desenvolvimento de software, o SCRUM pode ser benéfico para qualquer equipe que trabalhe em direção a um objetivo comum.

Dentro do SCRUM alguns dos itens que são trabalhados são:

a) US: *User stories* (histórias), são um tipo de *work item* (item de trabalho) utilizado nos times ágeis, no qual consistem as regras de negócios das funcionalidades que serão criadas ou alteradas em um sistema;

b) *Bug*: é um outro tipo de *work item*, que consiste em um erro ou falha que ocorre num sistema ou programa, resultando num comportamento incorreto, inesperado ou fora do pretendido.

c) Débito técnico: não é um *work item*, pode ser definido da seguinte forma: "Durante a execução de um projeto de *software*, se existem itens que se decide por não realizar, e que ao não realizá-los, se paralisa o desenvolvimento futuro do projeto, esses elementos produzem débito técnico" (CUNNINGHAM, 1992).

Em poucas palavras, os integrantes de um time que utiliza o SCRUM são:

a) *Product Owner* (PO): o qual cria e ordena as US's;

b) Scrum master (SM): são os que usam o planejamento de sprint para evitar tarefas de desenvolvimento que se sobrepõem aos limites da equipe, coordenam o status e o esforço entre as equipes e integram bases de código (BASS, 2014).

c) Time de desenvolvimento: os que participam de todo processo de desenvolvimento, teste e infraestrutura do software. Se dividem em Desenvolvedores (devs), que fazem a codificação, *testers* (QA), que fazem as validações e *DevOps* que garantem a infraestrutura para o software.

2.2.2 Metodologia Ágil

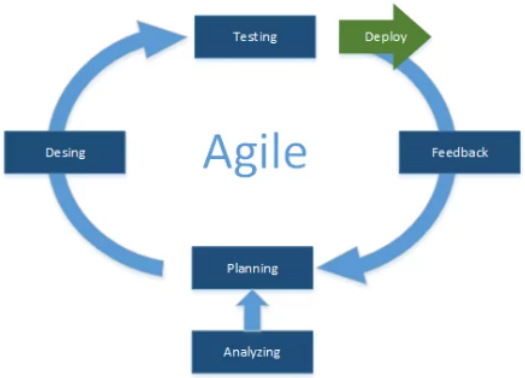
No contexto da área de tecnologia da informação, são altamente multifuncionais, com programadores, testadores e outros trabalhando lado a lado durante cada interação para que a qualidade possa ser construída nos produtos através de técnicas como desenvolvimento orientado a testes de aceitação, uma grande ênfase em testes automatizados e o pensamento/conhecimento de toda a equipe (CRISPIN; GREGORY, 2009). As equipes ágeis utilizam técnicas como refatoração e preferência pela simplicidade para evitar o acúmulo de débito técnico.

As principais práticas usadas por equipes ágeis estão relacionadas a testes. Os programadores ágeis usam o desenvolvimento orientado a testes (TDD), também chamado de design orientado a testes, para escrever código de produção de qualidade. Com o TDD, o programador escreve um teste para um pequeno pedaço de funcionalidade, o vê falhar, corrige o código para que o teste passe e então segue para a próxima parte da funcionalidade. (CRISPIN; GREGORY, 2009)

As equipes ágeis trabalham em estreita colaboração com o negócio e têm uma compreensão detalhada dos requisitos. Eles estão focados no valor que podem oferecer e podem ter uma grande quantidade de informações para priorizar recursos. (CRISPIN; GREGORY, 2009)

Na Figura 2 é possível entender como o agile é fluido e o trabalho do time é feito em paralelismo, ao invés de cascata.

Figura 2 – Modelo Ágil



Fonte: Adaptado de (DAVIES, 2018)

2.2.2.1.Manifesto ágil

O Manifesto Ágil é um documento que identifica quatro valores-chave e 12 princípios que seus autores acreditam que os desenvolvedores de software devem usar para orientar seu trabalho. O seu propósito é:

> "Estamos descobrindo melhores maneiras de desenvolver software fazendo-o e ajudando outros a fazê-lo. Valorizamos:
> - Indivíduos e interações sobre processos e ferramentas.
> - Software funcionando sobre documentação abrangente.
> - Colaboração do cliente sobre a negociação do contrato.
> - Responder à mudança em vez de seguir um plano." (FOWLER; HIGHSMITH, 2001).

A frase acima foi criada por um grupo de desenvolvedores de software experientes e reconhecidos, no qual indica que os membros realmente praticam métodos citados em seu próprio trabalho.

Esses desenvolvedores querem ajudar os outros com métodos ágeis, e aprofundar o próprio conhecimento aprendendo com aqueles que tentam ajudar. As declarações de valor têm uma forma: em cada marcador, o primeiro segmento indica uma preferência, enquanto o último segmento descreve um item que, embora importante, é de menor prioridade. Essa distinção está no cerne de agilidade e é reconhecida a importância do processo e das ferramentas, com o reconhecimento adicional de que a interação de indivíduos qualificados é de importância ainda maior (FOWLER; HIGHSMITH, 2001).

Da mesma forma, a documentação abrangente não é necessariamente ruim, mas o foco principal deve permanecer na entrega de software funcional. Portanto, cada equipe de projeto precisa determinar, por si mesma, qual documentação é absolutamente essencial.

No mundo turbulento de negócios e tecnologia, seguir escrupulosamente um plano pode ter consequências terríveis, mesmo que seja executado fielmente. Por mais cuidadosamente que um plano seja elaborado, torna-se perigoso negar mudanças. Durante a criação do manifesto, foram criados 12 princípios, apresentados na Figura 3.

Figura 3 – Princípios

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de *software* com valor agregado.	7. Software funcionando é a medida primária de progresso.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.	8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.	9. Contínua atenção a excelência técnica e bom design aumenta a agilidade.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.	10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
5. Construir projetos em torno de indivíduos motivados, dando a eles o ambiente e o suporte necessário e confiando neles para fazer o trabalho.	11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.	12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Fonte: (Autoria própria, 2021)

2.2.2.2. Agile Testing

As práticas de testes ágeis não se limitam a membros de equipes ágeis. Também podem ser usadas para melhorar a avaliação de projetos usando metodologias tradicionais de desenvolvimento.

O desenvolvimento ágil incentiva a resolver problemas como equipe. Pessoas de negócios, programadores, testadores, analistas – todos envolvidos no desenvolvimento – decidem juntos a melhor forma de melhorar seu produto. O melhor de tudo é que, como testadores, o trabalho é em conjunto com uma equipe de pessoas que se sentem

responsáveis por oferecer a melhor qualidade possível, focadas em testes (GREGORY; CRISPIN, 2014).

2.2.2.2.1. Testador ágil

Um testador ágil é definido como um testador profissional que aceita mudanças, colabora bem com pessoas técnicas e de negócios e entende o conceito de usar testes para documentar requisitos e impulsionar o desenvolvimento. Tendem a ter boas habilidades técnicas e comportamentais.

Estão dispostos a aprender o que os clientes fazem para que possam entender melhor os requisitos do software e tendem a ver o quadro geral, analisando o sistema do ponto de vista do usuário, o que significa que geralmente são focados no cliente.

2.2.2.2.2. A mentalidade de teste ágil

Uma equipe ágil é aquela que se concentra continuamente em fazer seu melhor trabalho e entregar o melhor produto possível. Para isso, é envolvida muita disciplina, aprendizado, tempo, experimentação e trabalho em conjunto (GREGORY; CRISPIN, 2014).

2.2.2.2.3. Aplicação de princípios e valores ágeis

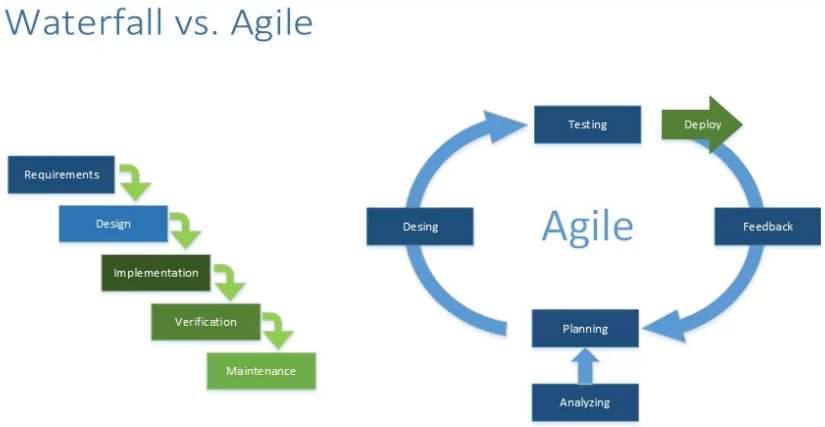
Os valores e princípios ágeis promovem um foco nas pessoas envolvidas em um projeto e como elas interagem e se comunicam, gerando maior moral de equipe e melhor velocidade do que uma equipe de indivíduos talentosos que funciona mal. Os princípios que são considerados importantes para um testador ágil são: fornecer *feedback* contínuo; entregar valor ao cliente; habilidade de comunicação cara a cara; coragem; manter simples; praticar a melhoria contínua; responder à mudança; organização; foco nas pessoas; curtir o trabalho.

Uma das contribuições mais importantes do testador ágil é ajudar o PO ou o cliente a articular os requisitos para cada US na forma de exemplos e testes. Trabalha em conjunto com os colegas de equipe para transformar esses requisitos em testes executáveis e junto com outros membros da equipe trabalham para executar esses testes cedo e com frequência, para que sejam continuamente guiados por *feedback*.

2.3. Comparativo dos métodos *Agile* vs *Waterfall*

Na Figura 4 é apresentado o fluxo da metodologia ágil e tradicional.

Figura 4 – Modelo cascata vs Modelo Ágil



Fonte: Adaptado de (DAVIES, 2018)

No diagrama de abordagem em fases, fica claro que o teste acontece no final, porém o diagrama é idealista, pois dá a impressão de que há tanto tempo para testar quanto para codificar. Em muitos projetos este não é o caso, o teste é "esmagado" porque a codificação leva mais tempo do que o esperado e as equipes entram em um ciclo de código e correção no final.

No agile, os testadores testam cada incremento de codificação assim que termina. Uma iteração pode ser tão curta quanto uma semana, ou tão longa quanto um mês. A equipe cria e testa um pouco de código, certificando-se de que funciona corretamente e, em seguida, passa para a próxima parte que precisa ser construída. Os programadores nunca chegam à frente dos testadores, porque uma US não está "pronta" até que seja testada, portanto a metodologia é iterativa e incremental (RIVAS; GODOY DE SOUZA, 2014).

3. Materiais e Métodos

A metodologia utilizada para o desenvolvimento do sistema proposto inicia-se na consolidação dos conhecimentos teóricos previamente apresentados e na definição dos requisitos e recursos a serem utilizados.

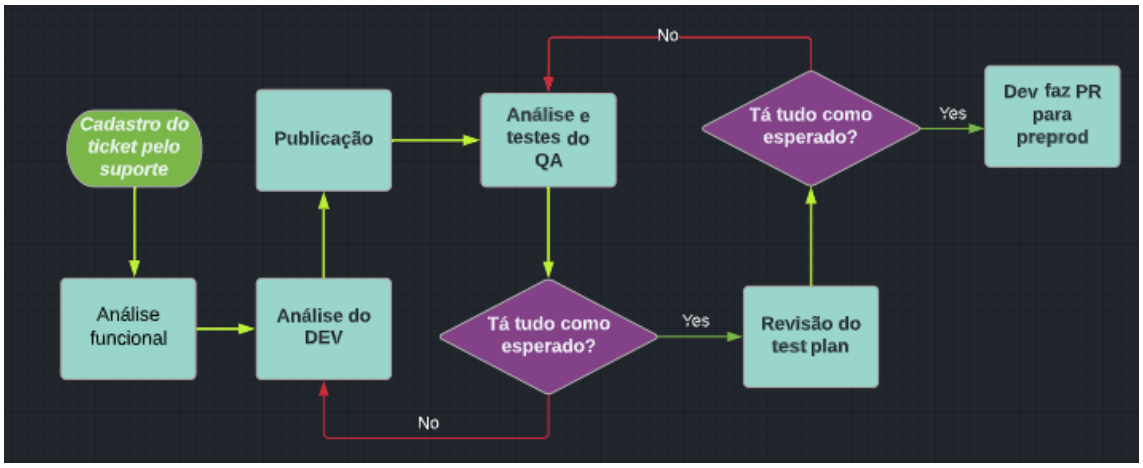
Em seguida, é feita uma demonstração teórica do porquê o *agile testing* pode ser a melhor escolha para o desenvolvimento de software a partir de um estudo de caso que será elaborado para uma empresa do setor de tecnologia da informação.

3.1. Fluxo do time antes da implementação

O time escolhido para ser implantado o *agile testing* trabalha apenas com os *bugs* reportados pelos clientes, ou seja, um time normalmente conhecido como sustentação, e nele era utilizado o modelo em cascata, previamente explicado.

Por conta dos curtos prazos de entrega das correções dos *bugs* de clientes e da grande quantidade de itens, o time não conseguia manter o QA e o *dev* trabalhando em paralelo, então o fluxo utilizado está representado na Figura 5.

Figura 5 - Fluxo antigo do time



Fonte: (Autoria própria, 2022)

Pelo fato de existirem clientes que precisam da correção dos *bugs* com mais rapidez, foi pensado em implantar o *agile testing* de uma forma que a cultura do time não fosse abruptamente afetada e com isso fosse possível manter as entregas sem atraso, até a adaptação do time com o novo fluxo.

3.2.Requisitos para implantação

3.2.1.Adaptação da metodologia

Para que a metodologia fosse adaptada para o time, era necessário ter o conhecimento dos requisitos funcionais do sistema fabricado pela empresa, das ferramentas da empresa e como utilizá-las a favor da implantação, da cultura do time e, claro, conhecimento da metodologia do *agile testing*. A partir de tudo isso, poderia ser iniciada a adaptação para a cultura do time, que será melhor descrita nos próximos tópicos.

3.2.2. Recursos para implantação

Para a implantação do *agile testing*, fez-se necessário uma longa conversa com a gestão para poder avaliar os riscos, como seria o novo fluxo e quais seriam as pessoas que iriam trabalhar nessa frente. Em seguida, fez-se necessário conversar e apresentar a ideia para alguns membros do time e assim averiguar se existia mais alguma linha falha para poder ajustar e por fim apresentar ao time como um todo.

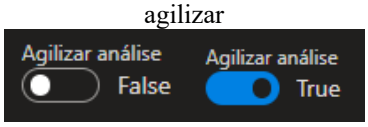
Ao realizar todos os ajustes necessários, no final de março de 2022 foi feita a apresentação e implantação do método proposto, que foi colocado em prática no início de abril de 2022.

3.2.3.Ferramentas utilizadas

Foi feita uma automação de mensagens através do *Power automate* para que quando um item fosse impactante à algum cliente ou dentro do time, seria enviado um e-mail ao QA responsável avisando que aquele item precisa de mais atenção (mais a frente será explicado qual foi o critério utilizado para definir se os itens eram ou não impactantes).

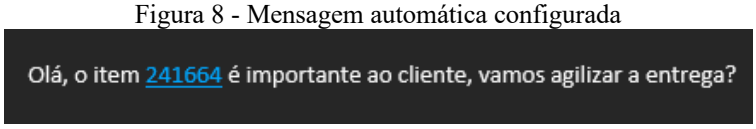
Para isso foi criado um botão dentro da ferramenta *Azure DevOps*, onde o time utiliza de *dashboards* para visualização dos itens. Esse botão fica dentro de cada *bug* e quando alguém percebe a necessidade de agilizar o item, o botão é apertado e um e-mail é enviado ao QA que irá agilizar a análise do item. Na Figura 6 é mostrado o botão e na Figura 7 a mensagem adaptada que é enviada por e-mail.

Figura 6 - Botão criado para agilizar



Fonte: (Autoria própria, 2022)

Figura 8 - Mensagem automática configurada

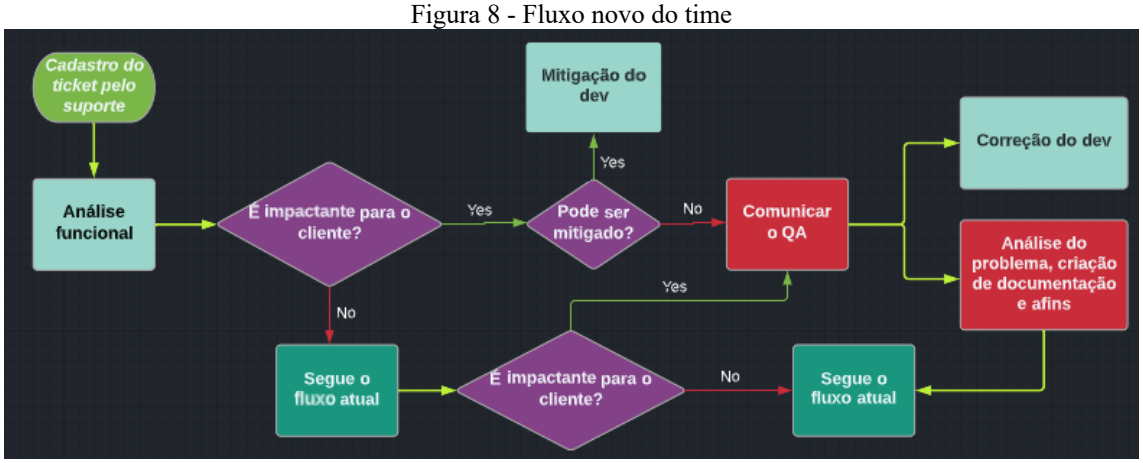


Fonte: (Autoria própria, 2022)

3.3. Implantação da metodologia adaptada

Visto que em nenhum momento o cliente poderia ser impactado com essa implantação, a metodologia não poderia ser completamente utilizada, portanto foi feito o esquema apresentado na Figura 8.

Figura 8 - Fluxo novo do time



Fonte: (Autoria própria, 2022)

Como pode ser visto na imagem, o QA será acionado quando um item for impactante para o cliente, e irá realizar: análises do problema do cliente e dos possíveis impactos, planejamento de testes e documentação. Isso tudo em paralelo com a correção do *bug* feita pelo *dev*. Nesse momento, ambos trabalham juntos em prol de buscar uma melhor solução, de acordo com as regras de negócios do sistema, tudo com o objetivo de não haver reabertura do item tanto no fluxo do próprio time, quanto pelo cliente.

A escolha de mudança de ação do QA trazendo-o para antes no fluxo, é prevista no *agile testing*, para poder realizar o planejamento, entender o problema e tentar minar qualquer impacto antes da correção ir para o ambiente de testes, gerando menos retrabalho e mais qualidade nas entregas.

Mas antes de chegar no QA e o botão de 'Agilizar análise' ser acionado, o *bug* é passado pelo PO do time, o qual irá analisar se o item realmente é um *bug* ou um comportamento esperado do sistema. Sendo *bug* o próprio PO já pode sinalizar se o item é impactante ou não para o cliente.

Após a análise do PO, o *dev* irá analisar o item, e se for notado que pode ser mitigado, o QA não será acionado. Se o item não for mitigável e o desenvolvedor também perceber a necessidade de agilizar o item, então o botão será marcado e com isso o QA sinalizado. Para saber se um item pode ou não ser mitigado, há a necessidade da análise do desenvolvedor, visto que a mitigação é feita sem alteração de código, não havendo qualquer mudança de comportamento esperado no sistema. Por esse motivo não há necessidade de passar pelo QA.

Porém, se, tanto na análise do PO quanto na análise do *dev* o item não for considerado impactante e nem mitigável, o item seguirá o fluxo antigo, no qual o *dev* irá corrigir, o ambiente do time irá receber a correção, e o QA irá: entender o problema; entender a solução do problema; analisar possíveis impactos; fazer o planejamento de testes; documentação dos testes; executar os testes manuais.

Caso o QA veja algum problema na correção, o item será retornado ao *dev* para que ele possa corrigir as questões encontradas.

A definição do que seria impactante ou não ao cliente é a chave para toda a implantação da metodologia no time, essa análise é feita da seguinte forma:

- Se o cliente for muito importante;
- Se tiver mais de 1 cliente reclamando do mesmo problema;
- Se o bug impedir que o cliente trabalhe;
- Se o item estiver há muito tempo na *board*;

Porém, pode ocorrer de o item entrar em uma dessas opções de impacto ao cliente durante o desenvolvimento do *dev*, então no meio do caminho o QA será acionado para agilizar a análise, pois o item se tornou antigo na *board*, ou vários clientes estão reclamando do mesmo problema.

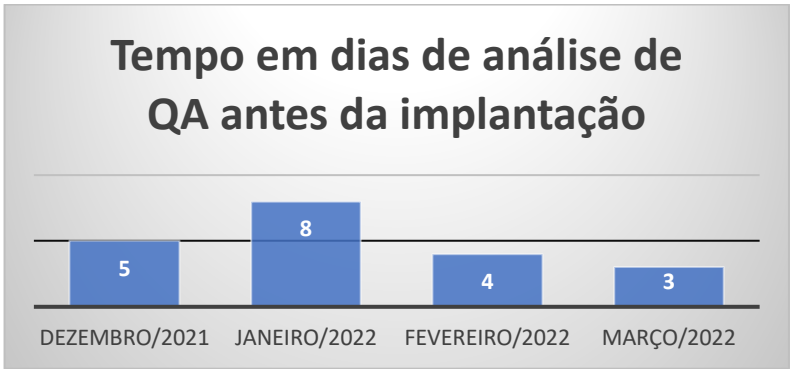
4. Resultados e Discussões

São apresentados a seguir os resultados obtidos na implantação da metodologia e discussões sobre os resultados e a evolução do processo.

4.1. Melhoria no tempo do fluxo

Como pode ser visto na Figura 9 o tempo médio de análise de um QA nos quatro meses antes da implantação era de 5 dias.

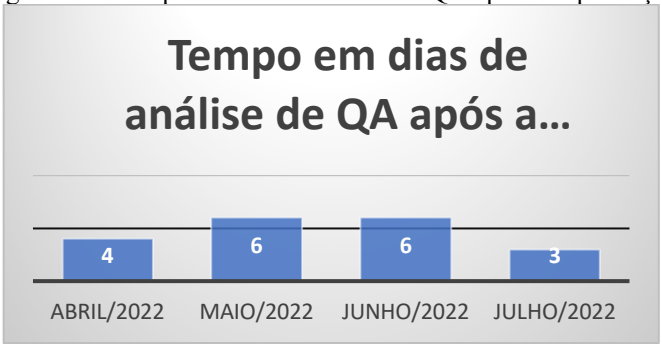
Figura 9 - Tempo em dias de análise de QA antes da implantação



Fonte: (Autoria própria, 2022)

Com a implantação do novo fluxo, o tempo reduziu-se basicamente pela metade, visto que o entendimento do problema, a análise dos possíveis impactos, o planejamento de testes e sua documentação já eram realizadas enquanto o dev realizava a correção, ou seja, no momento que chegava para o QA no fluxo, tinha-se apenas que averiguar se o que foi planejado se aplicava à correção do dev, caso necessário poderia ser feita uma melhoria no planejamento dos testes e eram realizados os testes manuais. Porém, como a sinalização estava ocorrendo tardiamente, a maioria dos itens acabaram mantendo no mesmo tempo anterior, como pode ser visto na Figura 10, em uma média de 4 a 5 dias.

Figura 10 – Tempo em dias de análise de QA após a implantação



Fonte: (Autoria própria, 2022)

4.2.Adaptação do time

Houveram alguns equívocos ao marcarem o botão de agilizar análise, pois em alguns itens a análise foi iniciada, porém a correção não foi realizada, havendo mitigação ou impossibilidade de reproduzir o problema reportado pelo cliente. Com isso, o QA realizava análise de um item desnecessariamente, gastando assim um tempo que poderia ter sido melhor aproveitado em outras tarefas para o time.

Outra questão que ocorreu foi os *devs* perceberem a necessidade de agilizar a análise, porém marcavam o botão tardiamente, e com isso o tempo de análise que seria salvo nos testes foi completamente perdido. Apesar dessas questões, o time inteiro se empenhou e adotou a metodologia, vendo um motivo e possível ganho no fluxo.

4.3.Redução de falhas

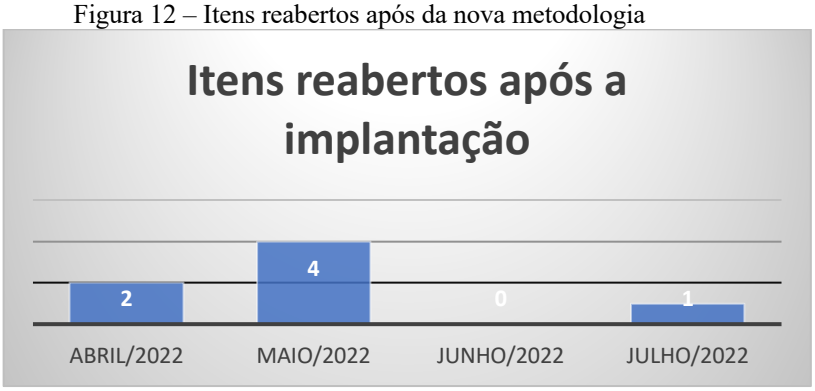
Na Figura 11 há uma demonstração dos itens que foram reabertos antes da implantação da nova metodologia.

Figura 11 – Itens reabertos antes da nova metodologia

Fonte: (Autoria própria, 2022)

Pelas questões relatadas no tópico anterior, a redução de itens reabertos quase não foi perceptível, pois não houve um trabalho paralelo entre a correção do desenvolvedor e a análise do QA. Na Figura 12 há uma demonstração dos itens que foram reabertos após a implantação da nova metodologia.

Figura 12 – Itens reabertos após da nova metodologia



Fonte: (Autoria própria, 2022)

4.4.Análise estatística

Foi realizada análise estatística em relação ao tempo e quantidade de reaberturas, antes e depois da implementação, como pode ser visto na Tabela 2

Tabela 2 - Análise estatística de dados

Implantação	Tempo de análise (dias)	Itens Reabertos
Antes	5	5
Antes	8	3
Antes	4	1
Antes	3	1
Depois	4	2
Depois	6	4
Depois	6	0
Depois	3	1

Fonte: (Autoria própria, 2022)

Para realizar as análises foi utilizado o Minitab e foram feitas 10 análises, 5 para tempo e 5 para quantidade de reaberturas em relação a implementação.

4.4.1.Tempo de análise (dias)

- Variáveis de Tempo de análise:

Figura 13 – Tempo VS Implementação

Statistics

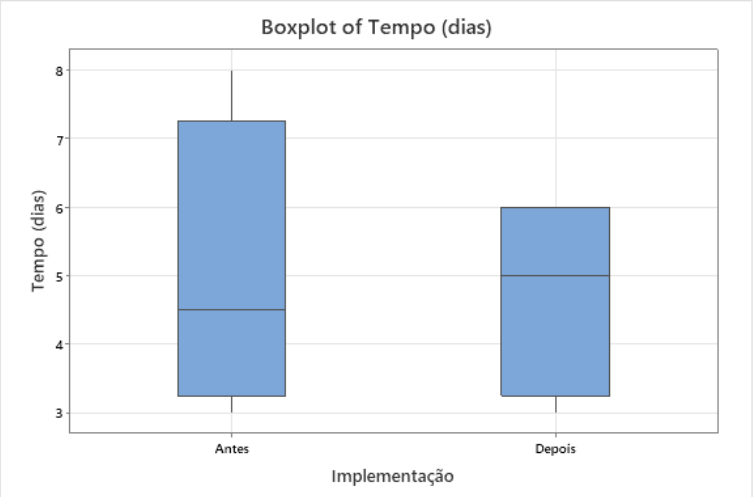
Variable	Implementação	N	N*	Mean	Minimum	Median	Maximum
Tempo (dias)	Antes	4	0	5.00	3.00	4.50	8.00
	Depois	4	0	4.750	3.000	5.000	6.000

Fonte: (Autoria própria, 2022)

A partir da Figura 13 apresentada, percebe-se que houve uma melhora no tempo de análise dos itens. Houve uma queda de 8 dias para 6 dias no tempo médio.

- BoxPlot do Tempo:

Figura 14 – Tempo VS Implementação



Fonte: (Autoria própria, 2022)

A Figura 14 comprova o resultado mostrado anteriormente de melhora no tempo de análise.

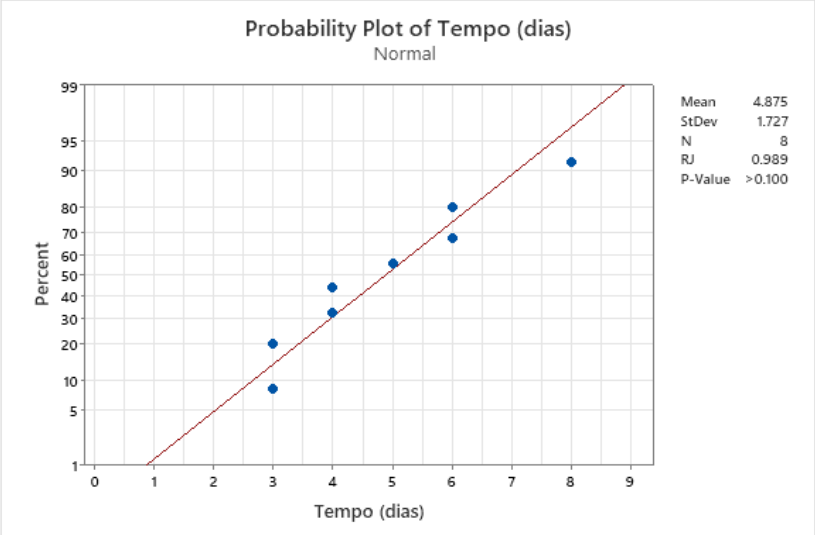
- Teste de normalidade

Foi escolhido o teste de Shapiro-Wilk por haverem apenas 8 elementos na amostra, que se referem à 4 meses antes da implantação e 4 meses após.

Para melhor entender o resultado dos valores abaixo:

- H0: os dados possuem distribuição normal;
- H1: os dados não possuem distribuição normal;
- O nível de significância para este teste é de 5% (0,05).

Figura 15 – Tempo VS Implementação



Fonte: (Autoria própria, 2022)

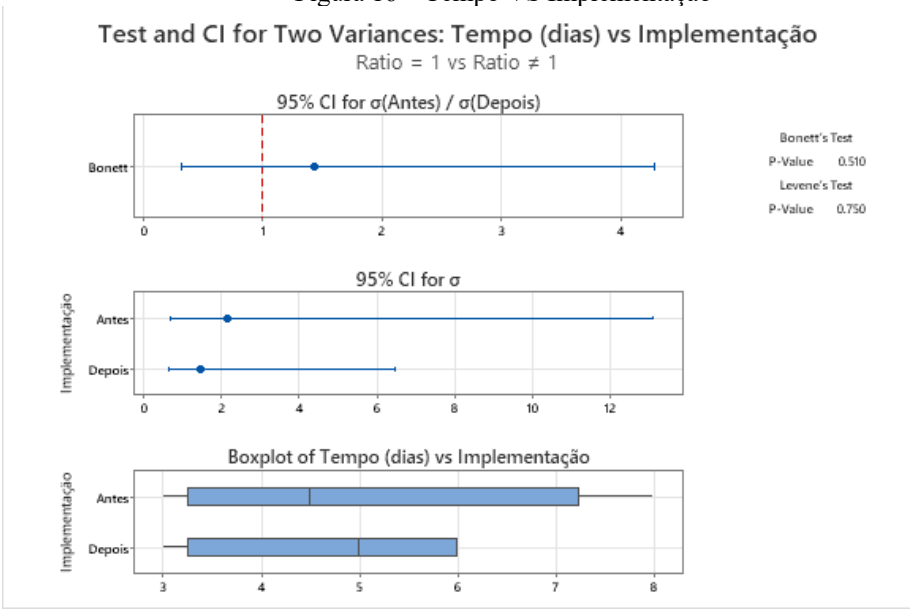
Visto que o p-value > 0,100 é superior ao nível de significância estabelecido de 5% (0,05) então aceita-se a hipótese nula de que os dados possuem uma distribuição normal. Como os dados possuem uma distribuição deve ser feito um teste de homocedasticidade (igualdade de variâncias).

- Teste de homocedasticidade (igualdade de variâncias)

  Para melhor entender o resultado dos valores abaixo:

  - H0: Os dados possuem homocedasticidade (igualdade de variâncias);
  - H1: Os dados não possuem homocedasticidade (igualdade de variâncias);
  - O nível de significância para este teste também é de 5% (0,05).

Figura 16 – Tempo VS Implementação



Fonte: (Autoria própria, 2022)

Pelo teste de Levene, como o p-value = 0,750 é superior ao nível de significância estabelecido de 5%, pode-se aceitar a hipótese nula de que os dados são homocedásticos (igualdade de variâncias). Como os dados possuem uma distribuição normal e também são homocedásticos, será utilizado o teste T para comparação das médias dos dois grupos.

- Comparação das médias

  Para melhor entender o resultado dos valores abaixo:

  - H0: médias da variável tempo são equivalentes entre o antes e depois da Implementação;
  - H1: médias da variável tempo são equivalentes entre o antes e depois da Implementação;
  - O nível de significância para este teste também é de 5% (0,05).

Figura 17 – Tempo VS Implementação

Test

Null hypothesis	$H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis	$H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
0.19	5	0.857

Fonte: (Autoria própria, 2022)

Visto que o p-value = 0,857 do tempo é superior ao nível de significância estabelecido de 5% (0,05), aceita-se a hipótese nula de que as médias são equivalentes, ou seja, não existe diferença significativa entre as médias dos dois grupos.

4.4.2.Reaberturas

- Variáveis de quantidade de Reaberturas:

Figura 18 – Reabertura VS Implementação

Statistics

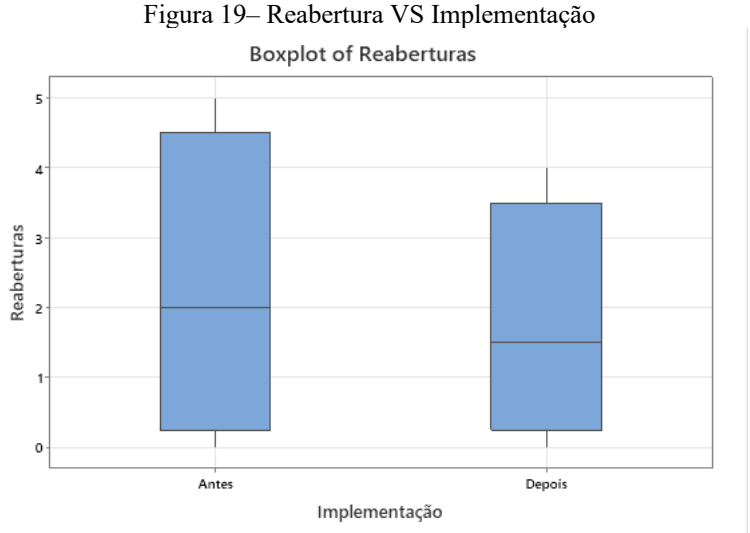
Variable	Implementação	N	N*	Mean	Minimum	Median	Maximum
Reaberturas	Antes	4	0	2.25	0.00	2.00	5.00
	Depois	4	0	1.750	0.000	1.500	4.000

Fonte: (Autoria própria, 2022)

Com a Figura 18 apresentada, percebe-se que houve uma melhora no número de reaberturas. Houve uma queda de 5 para 4 do número de itens reabertos.

- BoxPlot de Reaberturas:

Figura 19– Reabertura VS Implementação



Fonte: (Autoria própria, 2022)

A Figura 19 apresentada, comprova o resultado mostrado anteriormente de que houve uma melhora no número de reaberturas.

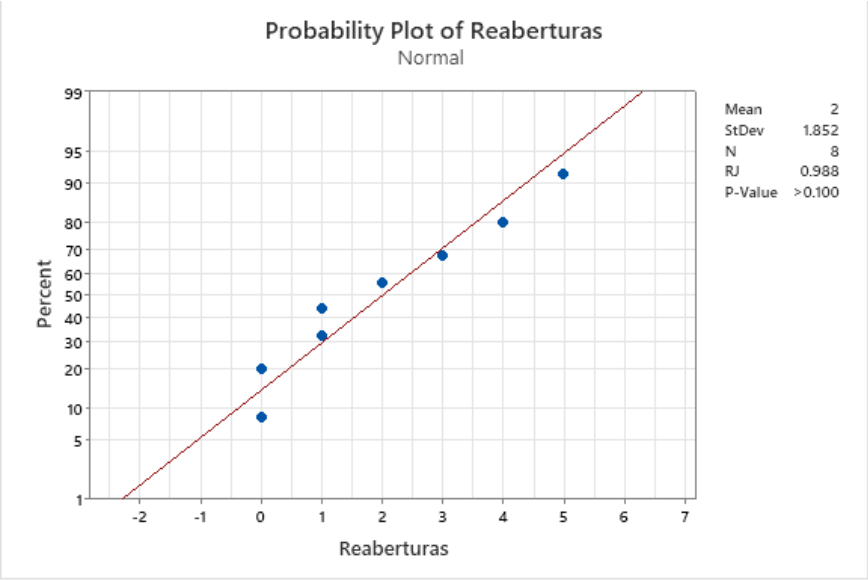
- Teste de normalidade

Foi escolhido o teste de Shapiro-Wilk por haverem apenas 8 elementos na amostra, que se referem à 4 meses antes da implantação e 4 meses após.

Para melhor entender o resultado dos valores abaixo:

- H0: os dados possuem distribuição normal;
- H1: os dados não possuem distribuição normal;
- O nível de significância para este teste é de 5% (0,05).

Figura 20 – Reabertura VS Implementação



Fonte: (Autoria própria, 2022)

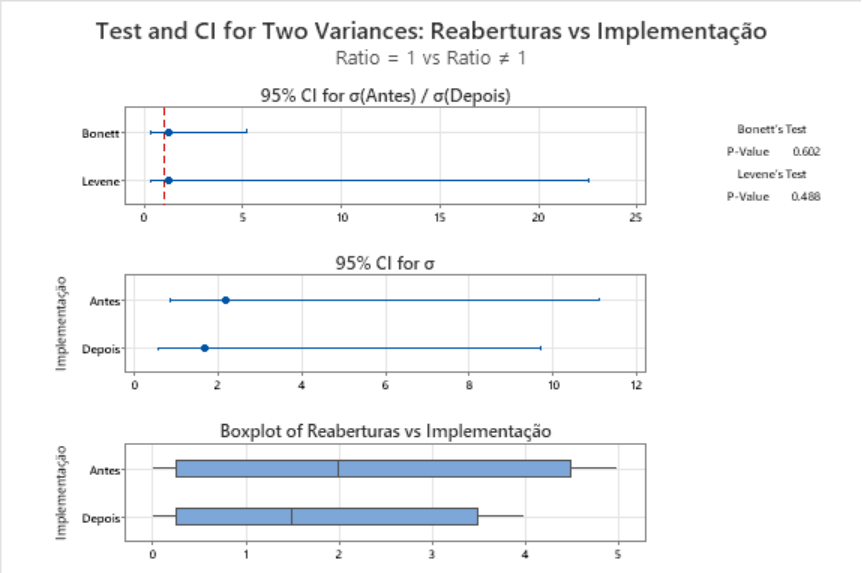
Visto que o p-value > 0,100 é superior ao nível de significância estabelecido de 5% (0,05) então aceita-se a hipótese nula de que os dados possuem uma distribuição normal. Como os dados possuem uma distribuição, deve ser feito um teste de homocedasticidade (igualdade de variâncias).

- Teste de homocedasticidade (igualdade de variâncias)

Para melhor entender o resultado dos valores abaixo:

- H0: Os dados possuem homocedasticidade (igualdade de variâncias);
- H1: Os dados não possuem homocedasticidade (igualdade de variâncias);
- O nível de significância para este teste também é de 5% (0,05).

Figura 21 – Reabertura VS Implementação



Fonte: (Autoria própria, 2022)

Pelo teste de Levene, como o p-value = 0,488 é superior ao nível de significância estabelecido de 5%, pode-se aceitar a hipótese nula de que os dados são homocedásticos (igualdade de variâncias). Como os dados possuem uma distribuição normal e também são homocedásticos, será utilizado o teste T para comparação das médias dos dois grupos.

- Comparação das médias

  Para melhor entender o resultado dos valores abaixo:

  - H0: médias da variável de reabertura são equivalentes entre o antes e depois da Implementação;
  - H1: médias da variável de reabertura são equivalentes entre o antes e depois da Implementação;
  - O nível de significância para este teste também é de 5% (0,05).

Figura 22 – Reabertura VS Implementação

Test

Null hypothesis	$H_0: \mu_1 - \mu_2 = 0$
Alternative hypothesis	$H_1: \mu_1 - \mu_2 \neq 0$

T-Value	DF	P-Value
0.36	5	0.735

Fonte: (Autoria própria, 2022)

Visto que o p-value = 0,735 de reabertura é superior ao nível de significância estabelecido de 5% (0,05), aceita-se a hipótese nula de que as médias são equivalentes, ou seja, não existe diferença significativa entre as médias dos dois grupos.

5. Conclusão

O projeto desenvolvido foi capaz de atingir parcialmente os objetivos propostos inicialmente, que foram estes a melhoria de tempo, redução de falhas e melhoria na qualidade, isso pode ser comprovado com os gráficos das análises realizados no tópico anterior.

5.1. Análise geral

Nesses 4 meses de análise dos dados coletados, foi concluído que, dos itens que foram analisados juntamente à correção do *dev*, houve uma redução de tempo e de falhas e melhoria na qualidade, porém é possível considerar que o resultado foi parcialmente atingido por conta dos problemas encontrados e relatados no tópico anterior.

Apesar dos números absolutos mostrarem que houve uma melhoria, a análise estatística mostra que esta melhoria não foi significativa, portanto outros estudos precisam ser feitos para comprovar essa questão.

A adaptação do time foi progressiva, contudo a resposta foi positiva, não havendo relutância em nenhum dos membros. Todos puderam perceber a vantagem de utilizar essa metodologia adaptada.

Porém foi notado que, apesar dos obstáculos, assim como toda metodologia, ainda há melhorias a serem feitas e mais adaptações a serem realizadas para que o time possa trabalhar com o *agile testing*.

5.2. Trabalhos Futuros

A implantação ainda está sendo utilizada dentro do time, pois a gestão e outros membros veem que há possibilidade de trabalhar com esse fluxo, apesar dos problemas encontrados. É um trabalho em desenvolvimento, estão sendo feitas melhorias nas questões citadas para que futuramente todos os itens possam ser tratados com essa metodologia e assim ter mais qualidade, agilidade e gerar mais satisfação ao cliente.

REFERÊNCIAS BIBLIOGRÁFICAS

BASS, Julian M. Scrum Master Activities: Process Tailoring in Large Enterprise Projects. *In*: 2014 IEEE 9TH INTERNATIONAL CONFERENCE ON GLOBAL SOFTWARE ENGINEERING 2014, **Anais** [...]. : IEEE, 2014. p. 6–15. DOI: 10.1109/ICGSE.2014.24. Disponível em: https://ieeexplore.ieee.org/abstract/document/6915249. Acesso em: 17 mar. 2022.

CRISPIN, Lisa; GREGORY, Janet. **Agile Testing: A Practical Guide For Testers And Agile Teams**. Illustrated ed. [s.l.] : Addison-Wesley Professional, 2009. Disponível em: http://index-of.co.uk/software-testing/agile_testing_-_a_practical_guide_for_testers_and_agile_teams.pdf. Acesso em: 7 jul. 2021.

CUNNINGHAM, Ward. The WyCash portfolio management system. *In*: ADDENDUM TO THE PROCEEDINGS ON OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS (ADDENDUM) - OOPSLA '92 1992, New York, New York, USA. **Anais** [...]. New York, New York, USA: ACM Press, 1992. p. 29–30. DOI: 10.1145/157709.157715. Disponível em: https://doi.org/10.1145/157710.157715. Acesso em: 17 jul. 2022.

DAVIES, Aran. **Waterfall Vs. Agile: Which Methodology Is Right For Your Project**. 2018. Disponível em: https://www.devteam.space/wp-content/uploads/2018/12/Waterfall-vs-Agile.png. Acesso em: 17 jun. 2022.

FOWLER, Martin; HIGHSMITH, Jim. The Agile Manifesto. **Software development**, *[S. l.]*, v. 9, n. 8, p. 28–35, 2001. Disponível em: http://www.awslad.com/wp-content/uploads/2010/01/The_Agile_Manifesto_SDMagazine1.pdf. Acesso em: 17 jul. 2022.

GREGORY, Janet; CRISPIN, Lisa. **More Agile Testing: Learning Journeys for the Whole Team**. 1. ed. [s.l.] : Addison-Wesley Professional, 2014. Disponível em: Http://Tisten.Ir/Wp-Content/Uploads/2019/04/More-Agile-Testing-Janet-Gregory-Lisa-Crispin-2014.Pdf. Acesso em: 6 jul. 2021.

MOHANTY, Hrushikesha; MOHANTY, J. R.; BALAKRISHNAN, Arunkumar. **Trends in Software Testing**. Singapore: Springer Singapore, 2017. DOI: 10.1007/978-981-10-1415-4. Disponível em: https://link.springer.com/chapter/10.1007/978-981-10-1415-4_7. Acesso em: 17 jul. 2022.

RIVAS, Mario Augusto; GODOY DE SOUZA, Enock. Análise comparativa da utilização do modelo tradicional (waterfall) de desenvolvimento de projetos e o modelo ágil (agile) em fábricas de software. **Revista de Sistemas e Computação**, Salvador, p. 3–11, 2014. Disponível em: https://core.ac.uk/reader/234559760. Acesso em: 1 maio. 2022.