UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

INTRODUZINDO MELHORIAS NOS PROCESSOS DE UMA EQUIPE DE TESTES DE REGRESSÃO: UM ESTUDO DE CASO

HELENA COUTRIN WADY

Orientador
Prof. Jobson Luiz Massollar da Silva

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2023

Catalogação informatizada pelo(a) autor(a)

W116 Wady, Helena Coutrin
INTRODUZINDO MELHORIAS NOS PROCESSOS DE UMA EQUIPE DE TESTES DE REGRESSÃO: UM ESTUDO DE CASO / Helena Coutrin Wady. -- Rio de Janeiro, 2023.
45 f.

Orientador: Jobson Luiz Massollar da Silva.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2023.

1. Teste de Regressão. 2. Automação de Teste. 3. Qualidade de Software. 4. Gerenciamento de Mudanças. I. Silva, Jobson Luiz Massollar da, orient. II. Título.

INTRODUZINDO MELHORIAS NOS PROCESSOS DE UMA EQUIPE DE TESTES DE REGRESSÃO: UM ESTUDO DE CASO

HELENA COUTRIN WADY

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado em: ___/___/_____

Banca examinadora:

_____

Prof. Jobson Luiz Massollar da Silva (Orientador)
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

_____

Prof. Gleison dos Santos Souza
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

_____

Prof. Reinaldo Viana Alvares
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2023

AGRADECIMENTOS

Agradeço aos meus pais, que sempre me apoiaram e acreditaram em mim, mesmo quando eu duvidei. Obrigada por vibrarem todas as minhas conquistas, mesmo que pequenas.

Agradeço à minha irmã, por estar sempre ao meu lado e por fazer os momentos difíceis se tornarem mais calmos.

Agradeço à minha madrinha, por ser meu exemplo de força e determinação.

Agradeço às minhas amigas, pelas palavras de encorajamento sempre que precisei.

Agradeço aos amigos que fiz na faculdade, por termos dividido (muitas) risadas e angústias. Obrigada por todo o apoio desde o início do curso, eu não teria conseguido sem vocês.

Agradeço imensamente ao professor Jobson, um dos professores mais excelentes que já conheci, que desde o início se prontificou a me orientar. Obrigada por todas as conversas e ensinamentos, sua mentoria foi essencial para o aperfeiçoamento deste trabalho.

Por fim, obrigada aos professores e colegas que fizeram parte desta jornada. Todos vocês contribuíram de alguma forma na minha trajetória como discente.

RESUMO

Um software passa por constantes mudanças durante seu ciclo de desenvolvimento, nas quais suas funcionalidades são alteradas e incrementadas visando atender aos requisitos definidos pelo cliente. Para avaliar se o sistema realmente cumpre os requisitos definidos são realizados testes de software. Existem tipos diferentes de testes de software, e, dentre eles, o foco deste trabalho é o teste de regressão. Ele tem como objetivo checar se uma funcionalidade continua operando corretamente após mudanças terem sido aplicadas no sistema. Por isso, é comum que esse tipo de teste seja automatizado em forma de *scripts* de teste. Para que os *scripts* de teste sejam efetivos, é essencial que as mudanças aplicadas no sistema sejam documentadas e repassadas, na forma e tempo adequados, para a equipe de testes. Nesse contexto, este trabalho tem como objetivo investigar como funciona o gerenciamento de mudanças dentro de uma equipe de teste de regressão, e propor possíveis melhorias a esses processos. Para isso, foram realizadas entrevistas com membros da equipe de teste de regressão, de diferentes cargos, para entender o contexto geral de trabalho. Por meio das informações capturadas nas entrevistas, foi possível elaborar melhorias a serem implementadas no processo atual de gerenciamento de mudanças da equipe. Foi possível entender também que existia um problema de comunicação da equipe de desenvolvimento com a equipe de teste de regressão para informar as mudanças implementadas no software, problema para o qual também foi sugerido um processo a ser seguido. A avaliação das melhorias propostas e do processo proposto foi feita por meio de entrevistas com os mesmos membros da equipe. Durante a avaliação foram comentados alguns pontos a serem corrigidos dentro das melhorias, mas o resultado dessa avaliação mostrou que a equipe concorda que as melhorias e o processo propostos de fato melhorariam o seu trabalho e são viáveis de serem implementadas dentro do time.

Palavras-chave: Teste de Regressão, Automação de Teste, Qualidade de Software, Gerenciamento de Mudanças

ABSTRACT

A software undergoes constant changes during its development cycle, in which its functionalities are changed and increased in order to meet the requirements defined by the client. To assess whether the system actually meets the defined requirements, software tests are performed. There are different types of software testing, and, among them, the focus of this work is regression testing. Regression testing aims to check whether a feature continues to operate correctly after changes have been applied to the system. Therefore, it's common for this type of testing to be automated in the form of test scripts. For the test scripts to be effective, it's essential that the changes applied to the system are documented and passed on, in the appropriate form and time, to the test team. In this context, this work aims to investigate how change management works within a regression test team and propose possible improvements to these processes. To this end, interviews were conducted with members of the regression test team, from different positions, to understand the general work context. Through the information captured in the interviews, it was possible to elaborate improvements to be implemented in the team's current change management process. It was also possible to understand that there was a communication problem between the development team and the regression test team to inform the changes implemented in the software, a problem for which a process to be followed was also suggested. The evaluation of the proposed improvements and the proposed process was done through interviews with the same team members. During the evaluation, some points to be corrected within the improvements were commented, but the result of this evaluation showed that the team agrees that the proposed improvements and process would in fact improve their work and are feasible to be implemented within the team.

Keywords: Regression Testing, Test Automation, Quality Assurance, Change Management

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Durante o ciclo de vida de um software, o sistema é constantemente modificado e incrementado com o objetivo de cumprir novos requisitos e aperfeiçoar o que já foi construído. Essas alterações, muitas vezes, acabam impactando negativamente em partes do sistema que já funcionavam em versões anteriores. Para avaliar se essas modificações introduziram defeitos em componentes novos e antigos do software e permitir a sua correção são realizados testes de software.

Diversos tipos de testes de software podem ser projetados e executados com o objetivo de avaliar diferentes características do sistema, como por exemplo funcionalidade, eficiência, usabilidade, segurança, dentre outras. Dentre os diversos tipos de testes de software existentes, os testes de regressão nos permitem avaliar se alterações no software geraram novos defeitos em funcionalidades previamente desenvolvidas e operantes.

Para executar um teste de regressão são criados casos de teste. O caso de teste exercita uma situação específica de uso com o objetivo de investigar se determinada funcionalidade do sistema está funcionando corretamente. Como os mesmos conjuntos de casos de teste geralmente são executados a cada ciclo de desenvolvimento, é recomendado que estes sejam executados de forma automatizada para manter a eficiência da execução dos testes (AMMANN e OFFUTT, 2017, p. 406). Para isso, existem ferramentas específicas que auxiliam no desenvolvimento e na execução de *scripts* de teste. O desenvolvimento desses *scripts* está baseado nos casos de teste e na documentação que é passada para a equipe responsável pelos testes.

Do ponto de vista da organização dos recursos humanos, muitas vezes as equipes de desenvolvimento e de testes são equipes distintas. Logo, para que os testes sejam desenvolvidos corretamente e ajustados quando necessário, a equipe de desenvolvimento deve documentar as mudanças que foram aplicadas ao sistema e repassá-las ao time de testes. Dessa forma, os testes permanecem alinhados com as modificações realizadas e são capazes de avaliar o software da forma adequada. Por outro lado, se as informações sobre as mudanças que foram aplicadas no software não forem repassadas à equipe de testes no momento e da forma apropriada, o processo

de testes tende a se tornar mais lento e sujeito a erros, impactando diretamente na liberação da nova versão do sistema.

1.2 OBJETIVOS

Esse trabalho tem como objetivo investigar, em um ambiente real, o processo de documentação e comunicação de mudanças que acontecem durante a fase de produção de um conjunto de sistemas e seus impactos na equipe responsável pela criação e execução dos testes de regressão. Para isso, será realizada uma pesquisa com pessoas que trabalham em uma empresa de T.I. (Tecnologia da Informação) para analisar como o gerenciamento de mudanças funciona na prática nesse ambiente, desde o seu tratamento pela equipe de desenvolvimento até os seus desdobramentos dentro da equipe de teste de regressão. A partir dos dados obtidos nessa pesquisa serão propostas mudanças para melhorar o processo existente e a integração das equipes de desenvolvimento e testes. Por fim, o processo proposto será avaliado para averiguar se ele é adequado e viável de ser implementado.

1.3 METODOLOGIA

As pesquisas necessárias para a condução deste trabalho serão realizadas por meio de entrevistas, caracterizando-se como uma pesquisa qualitativa e contendo também propostas de melhoria do processo atual.

Os entrevistados serão profissionais que trabalham em uma equipe de testes de regressão com diversos cargos (gerentes, desenvolvedores, líderes de equipe, etc.). As entrevistas serão conduzidas utilizando perguntas previamente elaboradas que pretendem investigar a estrutura da equipe, as ações que são executadas por ela quando ocorrem mudanças no sistema e a percepção dos entrevistados sobre os pontos fortes e fracos do processo vigente.

As informações fornecidas pelos entrevistados servirão de base para a criação do modelo de processo. A partir da análise desse processo e das demais informações coletadas, serão feitas propostas de melhoria.

Para realizar a avaliação das mudanças propostas buscou-se inspiração na metodologia TAM (Technology Acceptance Model - Modelo de Aceitação de Tecnologia) (DAVIS, 1989). TAM oferece um modelo de aceitação que busca entender quanto uma tecnologia é útil (percepção de utilidade) e quanto ela é fácil de usar (percepção da facilidade de uso) e essa avaliação é feita por meio de questionários. Neste trabalho, esses questionários serão substituídos por entrevistas de avaliação e a facilidade de uso será substituída pela viabilidade

(facilidade de implementação de uma mudança). Assim, no contexto desse trabalho, serão examinadas a percepção de utilidade das mudanças propostas (as mudanças trazem benefícios para a equipe?) e a percepção de viabilidade (é possível implementar essas mudanças no dia a dia da equipe?) sob a perspectiva da equipe de testes de regressão.

1.4 ORGANIZAÇÃO DO TEXTO

O presente trabalho está estruturado nos seguintes capítulos, além desta introdução:

Capítulo 2: Fundamentação teórica, onde estão os conceitos utilizados na elaboração do trabalho.

Capítulo 3: Identificação da situação atual, onde são apresentados os dados coletados por meio de entrevistas.

Capítulo 4: Propostas de melhorias, onde são apresentadas as mudanças no processo atual, bem como sua avaliação pela equipe.

Capítulo 5: Conclusão, apresenta as considerações finais, assinala as contribuições da pesquisa e sugere possibilidades de aprofundamento posterior.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos relacionados à qualidade de software e, em particular, aos testes de software. Outro ponto abordado relevante para este trabalho é o conceito de Gerenciamento de Mudanças e sua relação com os testes de regressão.

2.1 QUALIDADE DE SOFTWARE

A qualidade de software pode ser definida como "um processo efetivo de software aplicado de tal maneira que cria um produto útil que fornece valor mensurável para aqueles que o produzem e o utilizam." (PRESSMAN e MAXIM, 2020, p. 312, tradução do autor).

Uma das formas mais difundidas de avaliar a qualidade de um software é utilizando as características de qualidade propostas na ISO (2011), norma que fornece um modelo de referência na avaliação de softwares. Dentro desta norma estão definidos dois modelos de qualidade: modelo de qualidade em uso e modelo de qualidade de produto. O primeiro apresenta características que nascem da interação do usuário com o sistema em algum contexto específico, enquanto o segundo descreve características focadas em propriedades estáticas e dinâmicas de sistemas computacionais.

Segundo Pressman e Maxim (2020, p. 346-347), o modelo de qualidade em uso possui cinco características e suas respectivas definições:

- **Eficácia:** Exatidão e completude com qual os usuários alcançam objetivos.
- **Eficiência:** Recursos gastos para alcançar completamente os objetivos dos usuários com a exatidão desejada.
- **Satisfação:** Usabilidade, confiança, prazer, conforto.
- **Livre de riscos:** Mitigação de riscos econômicos, de saúde, segurança e ambientais.
- **Cobertura de contexto:** Completude, flexibilidade.

Além destas características, existem também as características do modelo de qualidade de produto:

- **Funcionalidade:** Completo, correto, apropriado.
- **Eficiência de desempenho:** Tempo, utilização de recursos, capacidade.
- **Compatibilidade:** Coexistência, interoperabilidade.

- **Usabilidade:** Adequação, aprendizagem, operabilidade, proteção contra erros, estética, acessibilidade.
- **Confiabilidade:** Maturidade, disponibilidade, tolerância a falhas, recuperabilidade.
- **Segurança:** Confidencialidade, integridade, responsabilização, autenticidade.
- **Manutenibilidade:** Modularização, reusabilidade, modificabilidade, testabilidade.
- **Portabilidade:** Adaptabilidade, instabilidade, substituibilidade.

Dentre essas características, o foco deste trabalho é a Funcionalidade. Essa característica é uma das mais importantes e representa o grau em que o produto ou sistema fornece funções que satisfaçam necessidades explícitas e implícitas quando utilizadas em condições especificadas.

Para tentar assegurar que um software tenha qualidade, além dos modelos de qualidade, diversas atividades relacionadas à garantia e ao controle de qualidade são executadas durante o processo de desenvolvimento. Algumas dessas atividades são: definição do processo de garantia da qualidade, gerenciamento de mudanças, revisões e testes de software, adoção de métodos e ferramentas, auditoria e conformidade, medição e comunicação, dentre outras (PRESSMAN e MAXIM, 2020, p. 340).

2.2 TESTES DE SOFTWARE

Os testes de software são elementos cruciais do processo de desenvolvimento. Para Sommerville (2011, p. 144), “o teste de software é destinado a mostrar que um programa faz o que é proposto a fazer e para descobrir os defeitos do programa antes do uso”. Bourque e Fairley (2014, p. 82), por sua vez, definem os testes de software como uma verificação dinâmica de que um programa proporciona comportamentos esperados em finitos conjuntos de casos de teste, os quais são adequadamente selecionados do domínio de execução - que geralmente é infinito. Os casos de teste são condições particulares de uso do software e definem o objetivo do teste, as pré-condições, os valores de entrada, as particularidades da execução e os valores esperados de saída. Normalmente, casos de teste que pertencem a um mesmo cenário são agrupados em suítes de teste.

Testar software é uma função de controle de qualidade que tem como objetivo principal detectar falhas e permitir a correção dos defeitos antes da entrega. Nesse cenário, o papel da garantia da qualidade do software é assegurar que os testes são planejados adequadamente e

conduzidos de forma eficiente, para que tenham maior probabilidade de atingir seu objetivo principal. (PRESSMAN e MAXIM, 2020, p. 342, tradução do autor).

Assim como existem diferentes perspectivas de qualidade oferecidas pelos modelos de qualidade, existem tipos distintos de testes que visam avaliar diferentes aspectos de qualidade da aplicação. Segundo Bourque e Fairley (2014, p. 84-86), os testes podem ser aplicados em dois níveis diferentes durante o processo de desenvolvimento e manutenção do software: baseado no alvo (o que será testado) e baseado no objetivo (propósito do teste). Em relação ao alvo do teste, este pode ser um módulo do sistema, um conjunto de módulos ou até o sistema inteiro. Assim, os autores categorizam três alvos distintos: unidade, integração e sistema, e os definem como:

- **Teste de Unidade:** verifica o funcionamento, de forma isolada, de elementos do software que podem ser testados separadamente. Normalmente o teste acontece com acesso ao código que está sendo testado e utilizando ferramentas de depuração (ferramentas que auxiliam a encontrar defeitos no código a partir das falhas detectadas nos testes).
- **Teste de Integração:** é o processo de verificar as interações entre os componentes do software. O teste de integração é geralmente uma atividade contínua em cada fase de desenvolvimento do sistema, na qual os engenheiros de software abstraem as perspectivas de nível inferior e focam na perspectiva da fase que eles estão integrando no momento.
- **Teste de Sistema:** preocupa-se em testar o sistema como um todo. Testes de unidade e de integração efetivos terão identificado muitos dos defeitos do software. O teste de sistema geralmente é considerado adequado para avaliar os requisitos não funcionais do sistema - como segurança, agilidade, precisão e confiabilidade.

Além do alvo do teste, Bourque e Fairley (2014) apresentam definições para os diferentes objetivos do teste, que visam analisar diversas características do sistema. Esses objetivos variam de acordo com o alvo do teste, explicitado anteriormente.

2.3 TESTES DE REGRESSÃO

O foco deste trabalho são os testes de regressão, que podem ser definidos como:

> Reteste seletivo de um sistema ou de seus componentes para verificar que modificações não causaram efeitos não intencionais e que o sistema/componente ainda atende aos requisitos especificados. Envolve selecionar, minimizar e/ou

> priorizar um subconjunto de casos de testes em uma suíte de teste existente. Testes de regressão podem ser executados em cada um dos níveis descritos anteriormente (Unidade, Integração ou Sistema), e também podem ser aplicados a testes funcionais e não funcionais (BOURQUE e FAIRLEY, 2014, p. 87).

Testes de regressão são particularmente importantes em cenários de Desenvolvimento Contínuo, onde novas versões de um sistema são entregues periodicamente e, por isso, é preciso garantir que as funcionalidades já existentes não foram afetadas pela inclusão de novas funcionalidades.

Uma abordagem simples para os testes de regressão consiste em reexecutar todos os casos de testes desenvolvidos para versões anteriores do sistema. Porém, mesmo essa abordagem de "testar tudo novamente" pode trazer problemas: casos de teste de versões anteriores podem não ser executáveis na nova versão e a reexecução de todos os casos de teste pode ter alto custo sem trazer os benefícios esperados.

Para Ammann e Offutt (2017, p. 407), caso um teste de regressão falhe, o primeiro passo deve ser averiguar se o "culpado" é a suíte de teste ou as mudanças que ocorreram no software - em ambos os casos, os testadores terão trabalho adicional a fazer. Mesmo que não haja falhas, ainda há trabalho a ser realizado, pois uma suíte de teste que foi executada sem nenhum erro na versão atual do software não necessariamente irá funcionar sem erros na próxima versão.

2.4 AUTOMAÇÃO DE TESTES

A execução manual de casos de teste é possível. Contudo, como a quantidade de casos de teste a serem executados tende a ser grande, essa forma de execução torna-se inviável, principalmente por conta da necessidade de reexecução dos testes. Nesse contexto, Ammann e Offutt (2017, p. 406) recomendam que os testes de regressão devem ser automatizados, ou seja, implementados em forma de *scripts* de teste e executados de forma automática.

Para automatizar os testes existem ferramentas específicas que auxiliam no seu desenvolvimento e execução, tornando essa atividade mais eficiente. Por meio dessas ferramentas são desenvolvidos os *scripts* de teste. Os *scripts* de teste são códigos criados para testar os cenários especificados pelos casos de teste (um exemplo de caso de teste seria "Realizar uma compra de um produto com pagamento via Cartão de crédito"). Com a criação desses *scripts*, a reexecução de um ou mais cenários, quando necessário, torna-se mais ágil. Existe, além disso, a possibilidade de reaproveitamento de *scripts* quando os cenários são

semelhantes, pois não há necessidade de implementar o mesmo código mais de uma vez. Dessa forma, o desenvolvimento de novos cenários de teste também fica mais simplificado.

Segundo Ammann e Offutt (2017, p. 61), o nível de dificuldade para criação dos *scripts* de teste em relação à programação depende do software que está sendo testado. Alguns testes podem ser automatizados com conhecimentos básicos de programação, mas existem também testes que requerem mais conhecimentos e habilidades. Nestes casos, o desenvolvedor de teste terá que simular condições específicas, adicionar outros softwares para acessar ou simular um hardware específico e controlar o estado do ambiente de testes, caso necessário.

2.5 GERENCIAMENTO DE MUDANÇAS

Sobre o gerenciamento de mudanças, Sommerville (2011) destaca que:

> O processo de gerenciamento de mudanças inicia-se quando um 'cliente' preenche e envia uma solicitação de mudança (...) que descreve a mudança requerida para o sistema. Pode tratar-se de um relatório de *bug*, em que são descritos os sintomas de um *bug*, ou uma solicitação para adicionar funcionalidade extra ao sistema (SOMMERVILLE, 2011, p. 478).

Pezze e Young (2008, p. 449) recomendam que uma suíte de teste de boa qualidade deve ser mantida durante as versões de um software. Assim, para manter a qualidade das suítes de teste e para que os testes continuem cumprindo seu papel principal, é indispensável que haja um bom gerenciamento das mudanças no projeto, pois essas mudanças vão influenciar diretamente nos testes do software.

Ammann e Offutt (2017, p. 408) classificam os tipos de mudança em um software como:

- **Corretiva**: um defeito é corrigido.
- **Perfectiva**: alguns aspectos de qualidade do software são aperfeiçoados sem mudar seu comportamento.
- **Adaptativa**: o software é alterado para funcionar em um ambiente diferente.
- **Preventiva**: o software é alterado para evitar problemas futuros sem mudar seu comportamento.

Ainda segundo Ammann e Offutt (2017), todos esses tipos de mudanças requerem a execução de um teste de regressão. Mesmo quando a funcionalidade do sistema não é alterada, as suítes de teste precisam ser reanalisadas para checar se elas permanecem adequadas com a versão mais atual do software.

As mudanças implementadas na nova versão do software podem impactar o formato de entradas e saídas, e os casos de teste podem não fazer mais sentido sem as mudanças correspondentes/necessárias, como apresentado por Pezze e Young (2008). Além disso, alguns casos de teste podem se tornar obsoletos pelo fato de testarem funcionalidades do software que foram modificadas, substituídas ou removidas da nova versão.

Dessa forma, para que os testes sejam planejados, desenvolvidos e executados corretamente é primordial que as alterações feitas no software sejam registradas e comunicadas à equipe de testes. Esses registros devem conter as informações necessárias para que a equipe de testes tenha conhecimento do que foi alterado - seja essa alteração a inclusão de uma nova funcionalidade ou a mudança de uma já existente - e para que consigam ajustar o que for necessário nos casos de teste e nos *scripts* de teste, de forma que os testes permaneçam compatíveis com a versão mais atual do sistema.

Apesar da importância do gerenciamento de mudanças, os responsáveis pelo desenvolvimento do sistema nem sempre registram e comunicam as mudanças de forma adequada. Essa má comunicação pode levar os *scripts* de teste a não refletirem a versão mais atualizada do software e, consequentemente, falharem durante sua execução no período de teste de regressão. Quando isso acontece, a equipe de testes pode executar o caso de teste manualmente e, após o término do teste de regressão corrigir o *script* ou parar durante o teste de regressão para corrigir o *script* - em ambos os casos existe um retrabalho e, consequentemente, maior tempo gasto para terminar a execução do cenário completo de testes. Isso pode impactar na execução de outros cenários e também em métricas que são apresentadas ao cliente no final de cada teste de regressão, como a porcentagem de automação, por exemplo.

3 IDENTIFICAÇÃO DA SITUAÇÃO ATUAL

Esse capítulo apresenta as atividades relacionadas à identificação da situação atual do processo de mudanças nos sistemas sob a perspectiva da equipe de testes, como essas atividades foram conduzidas e seus resultados. Para isso foram realizadas três entrevistas com membros de diferentes funções. A primeira entrevista foi realizada com o líder de automação, a segunda com a arquiteta de teste e a terceira com os sub-líderes das *squads*. O objetivo dessas entrevistas foi entender o processo de mudanças nos sistemas e questionar se existiam possíveis melhorias a serem aplicadas nele.

A equipe de testes de regressão que participou deste estudo possui três atribuições principais no seu dia a dia de trabalho: ela é responsável por estudar os novos cenários de teste que serão desenvolvidos, desenvolver os *scripts* de teste (para testar estes cenários) e executar os *scripts* durante o período de teste de regressão. O teste de regressão, dentro deste projeto, acontece todo mês e normalmente tem um período de duração de 10 dias úteis.

3.1 PESQUISA PRELIMINAR

A investigação inicial de informações sobre o processo atual foi feita por meio de uma entrevista com o líder de automação de testes. Essa entrevista foi feita por videochamada e durou cerca de 40 minutos. A entrevista foi conduzida com o uso de um questionário disponível no Apêndice A.

3.1.1 Material Utilizado na Pesquisa

O questionário da pesquisa preliminar foi preparado antecipadamente e é composto por uma série de perguntas abertas organizadas em blocos de assuntos. O principal objetivo das perguntas era entender como as mudanças nos sistemas chegavam até a equipe de testes e como ela tratava essas mudanças. Todas as perguntas contribuíram para entender o ciclo de vida das mudanças: como elas eram criadas e documentadas, como eram repassadas para a equipe de testes e como/quando eram finalizadas (entregues).

3.1.2 Resultado da Pesquisa

O primeiro bloco de questões da entrevista buscou identificar a estrutura da equipe. Com as respostas dadas pelo entrevistado, constatou-se que a equipe possui 31 membros e está dividida em 5 *squads* (grupos/subequipes), cada uma para um tipo de sistema diferente: e-

commerce, mobile, faturamento, CRM (*Customer Relationship Management*) e demais sistemas. Os papéis desempenhados pelos membros são de estudo (encarregado de estudar o cenário e criar o documento do cenário de teste - este documento possui o passo a passo que o *script* de teste deverá realizar), automação de testes (encarregado de desenvolver os *scripts* de teste) e execução de testes (encarregado de executar os *scripts* de teste durante o período de teste de regressão). Atualmente, a maioria dos membros possui duas funções, como por exemplo estudo e execução de testes, bem como automação e execução de testes. Dentro de cada *squad* existem integrantes seniores, juniores e estagiários, e cada *squad* possui seu sub-líder.

No segundo bloco de questões, as perguntas visaram entender como a comunicação de mudança chega à equipe de testes. Foi apurado que esta comunicação acontece por meio de e-mails. A arquiteta de teste recebe um e-mail do cliente com as mudanças a serem aplicadas para o próximo teste de regressão (por exemplo, a inclusão de um novo cenário de teste) e avalia se essa mudança é viável para ser incluída nele. Este novo cenário de teste é criado para testar uma nova funcionalidade do sistema, e essa nova funcionalidade é apresentada pelos desenvolvedores em uma reunião semanal de mudanças. A viabilidade da criação do cenário de teste é testada pela própria arquiteta, que o executa manualmente e verifica se o fluxo está correto e funcional (às vezes uma funcionalidade pode estar não terminada, impossibilitando o andamento do cenário - caso aconteça, o cenário é descartado até que esta seja concluída). Caso a mudança seja viável, ocorre um estudo do cenário de teste, feito pela parte da equipe de teste que é responsável pelos estudos e uma estimativa de complexidade e tempo necessário para o estudo e para o desenvolvimento, feita pelos sub-líderes.

Ao adentrar no terceiro bloco de questões, o foco destas era identificar como a equipe de testes trata a comunicação de mudanças. Após a comunicação da mudança, esta percorre o seguinte caminho: a arquiteta de testes comunica a mudança à equipe de estudos, a equipe de estudos comunica aos sub-líderes das *squads* e esses sub-líderes, por sua vez, passam as informações para os desenvolvedores de teste de suas respectivas *squads*. A equipe de estudos utiliza uma planilha compartilhada com os sub-líderes para informar o status do estudo do cenário de teste. A equipe de estudos cria o documento que descreve o novo cenário de teste que será desenvolvido e, quando o documento é terminado, o envia para a aprovação do cliente. Este documento contém o passo a passo que o *script* de teste deverá implementar (e.g., que campo ou botão deverá ser clicado, o que deverá ser digitado no campo de texto) e é

imprescindível para o desenvolvimento. Após a aprovação, o documento é colocado na planilha compartilhada e está pronto para ser utilizado no desenvolvimento dos *scripts* de teste.

O controle interno realizado pela equipe de teste e a delegação de tarefas para desenvolver o cenário de teste mapeado no documento são feitos em uma ferramenta de gestão de projetos chamada Jira[1]. O controle do desenvolvimento do cenário de teste é feito pelo cliente utilizando uma ferramenta de gestão de projetos chamada Rally (o líder do projeto cadastra nela as mesmas tarefas que foram criadas no Jira). Além disso, o monitoramento da execução dos cenários de teste durante o período de testes de regressão é feito pelo cliente utilizando a ferramenta de gestão de testes chamada ALM (*Application Lifecycle Management*). Após receberem o documento do cenário de teste, os sub-líderes criam o novo cenário na ALM (cada cenário, dentro desta ferramenta, é uma suíte de teste) e planejam os *cards* de tarefas no Jira. Esses *cards* são distribuídos entre os desenvolvedores do time para que cada um desenvolva uma parte do cenário. Os desenvolvedores dos *scripts* de teste estudam a documentação do cenário e criam os *scripts* de teste conforme especificados nos *cards*.

O quarto bloco de questões permeava apenas uma questão: o que a equipe de testes deve entregar? A resposta obtida foi que ela deve desenvolver os *scripts* de teste para testar os cenários, assim como registrar as suas execuções na ferramenta de gestão de testes do cliente (ALM). Durante os testes de regressão, também se espera que a equipe guarde as evidências das execuções de testes, o que normalmente é feito automaticamente pela ferramenta utilizada para executar os *scripts* de teste. Esta ferramenta, chamada UFT (*Unified Functional Testing*), realiza a captura (*print*) de cada tela durante a execução do *script* e gera um documento contendo todas elas. Caso o *script* falhe por não estar adequado ao funcionamento do sistema e o cenário seja executado manualmente, é necessário que o executor capture cada tela e as guarde em um documento.

Ao final da entrevista, foi perguntado ao entrevistado se ele possuía alguma observação a acrescentar e ele achou interessante pontuar que uma equipe de testes nestas características possui três atividades principais a serem realizadas: estudo e planejamento dos cenários, desenvolvimento dos *scripts* e execução dos casos de teste planejados.

3.1.3 Análise

[1] https://www.atlassian.com/software/jira

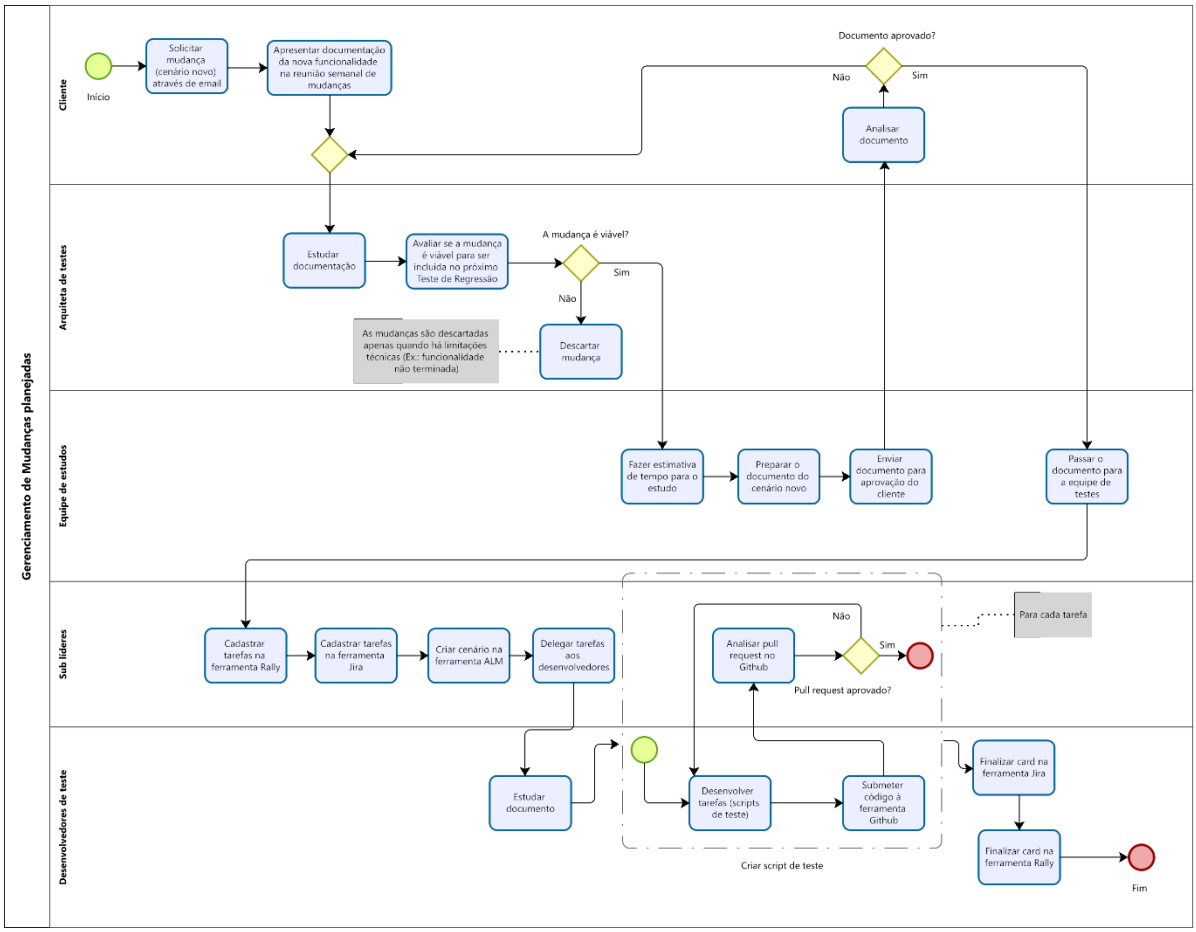
Com a pesquisa preliminar foi possível entender como a equipe está estruturada e como o ciclo de vida da mudança acontece sob a perspectiva da equipe de testes de regressão. Também foi possível entender que existem dois tipos de mudanças: mudanças planejadas e mudanças não planejadas.

As mudanças planejadas são os cenários novos, por exemplo, e são as mudanças que seguem o processo explicado anteriormente pelo entrevistado (são requisitadas pelo cliente, estudadas e os *scripts* de teste são desenvolvidos e executados).

Por outro lado, as mudanças não planejadas são, geralmente, alterações realizadas em funcionalidades já implementadas no software, não seguem o fluxo descrito anteriormente e requerem uma ação imediata por parte da equipe de testes durante o período de testes de regressão. Um exemplo deste tipo de mudança é a alteração da localização de um botão em uma página Web (um botão que fica no lado esquerdo passa para o lado direito). Este tipo de alteração pode fazer com que o *script* de teste não funcione mais e necessite de ajustes.

A Figura 1 apresenta o modelo BPMN (OMG, 2023) do processo atual de mudanças planejadas, elaborado de acordo com as informações dadas pelo líder de automação nessa entrevista.

Figura 1 - Processo de Gerenciamento de Mudanças Planejadas



Fonte: Elaborado pelo autor

A partir das informações coletadas e, principalmente, da descoberta de que existem dois processos diferentes de mudanças, decidiu-se entrevistar o ponto focal do recebimento da requisição da mudança (arquiteta de teste) para analisarmos como funciona o processo de mudança não planejada.

3.2 SEGUNDA PESQUISA

A segunda investigação foi feita com a arquiteta de testes da equipe, também por meio de uma entrevista. A entrevista foi realizada por videochamada e durou cerca de 30 minutos. Nesta entrevista, diferentemente da anterior, não havia um questionário previamente elaborado e o intuito era entender o processo de mudanças não planejadas.

3.2.1 Material Utilizado na Pesquisa

Nessa entrevista, o material utilizado foi o modelo em BPMN do processo de mudanças planejadas (Figura 1).

3.2.2 Resultado da Pesquisa

A primeira pergunta feita nesta entrevista, para conseguir entender o cenário de mudanças de forma mais ampla, foi se existia um processo a ser seguido em uma situação de mudanças não planejadas. A arquiteta relatou que não existia um processo a ser seguido, nem ações previamente estabelecidas a serem tomadas nessa situação. As mudanças são consideradas "não planejadas" por serem comunicadas sem a devida antecedência, ou seja, na verdade elas são planejadas, mas o problema está na comunicação que não acontece como deveria. A arquiteta de teste recebe um e-mail informando sobre a mudança somente quando esta se refere à inclusão de um cenário novo (para testar uma nova funcionalidade), mas quando a mudança é uma alteração em alguma funcionalidade já existente, esta comunicação não acontece. Dessa forma, a equipe de testes toma ciência dessas mudanças somente durante o período de teste de regressão, ao executar os testes com a versão mais atualizada do sistema. Assim, a carga de trabalho durante este período aumenta consideravelmente, pois além de corrigir os *scripts* que falham durante a execução, a equipe precisa executá-los antes que o período acabe.

Também foi mencionado pela entrevistada que não existe um canal de comunicação direto entre o time de desenvolvimento e o time de teste de regressão para informar as mudanças que são implementadas - as equipes de teste e de desenvolvimento são de empresas diferentes. Ocasionalmente, ao participarem de alguma reunião com os desenvolvedores, a arquiteta de teste e os sub-líderes tomam ciência de que haverá uma mudança em uma funcionalidade já existente, mas não possuem os detalhes necessários para aplicar as devidas correções nos *scripts* de teste.

Além disso, a entrevistada também comentou que existem algumas ações alternativas a serem tomadas a fim de que as informações de mudanças cheguem até o time de testes - ambas ainda não são utilizadas atualmente, mas estão sendo cogitadas. A primeira ação comentada foi ir em busca das informações de mudanças em uma ferramenta de gestão de projeto utilizada pelo cliente (Rally). Nesta ferramenta, a equipe de desenvolvimento cria *cards* com as informações de novas funcionalidades que serão implementadas e estes *cards* ficam organizados dentro de uma *sprint*. Desta forma, tendo acesso à ferramenta, o time de testes de regressão conseguiria coletar as informações de mudanças e se organizar melhor para adequar os *scripts* de testes às novas funcionalidades, após elas serem implementadas.

A segunda ação mencionada pela arquiteta foi colocar pelo menos um integrante da equipe de testes nas reuniões de início de *sprint* da equipe de desenvolvimento. Nestas reuniões acontecem o planejamento da *sprint* que irá iniciar, e nela são discutidas as novas funcionalidades que serão implementadas. Desta forma, tendo ao menos um integrante da equipe de testes participando ativamente dessas reuniões, as informações de mudanças poderiam ser coletadas e levadas para a equipe de teste com antecedência, fazendo com que as adaptações dos *scripts* de teste fossem realizadas em tempo hábil - após as mudanças serem desenvolvidas e antes do início do período de teste de regressão.

3.2.3 Análise

A partir da entrevista com a arquiteta de testes foi possível entender melhor a situação atual da equipe de testes frente às mudanças que ocorrem no sistema. Utilizando como ponto de partida as informações citadas pela entrevistada para contornar o problema da comunicação entre as equipes de desenvolvimento e de testes de regressão, entende-se que é possível propor um processo que consiga preencher as lacunas de comunicação e que tente garantir que as informações de mudanças nos sistemas cheguem ao time de testes de regressão em tempo hábil. Contudo, antes de começar a elaboração deste novo processo foi realizada uma última entrevista com os sub-líderes das *squads*.

3.3 TERCEIRA PESQUISA

Essa entrevista foi realizada com os sub-líderes das *squads* e tinha como objetivo entender quais melhorias eles julgam ser pertinentes no processo atual de mudanças planejadas.

3.3.1 Material Utilizado na Pesquisa

O material utilizado nessa entrevista foi o modelo BPMN de mudanças planejadas utilizado na segunda entrevista (Figura 1) e as perguntas foram feitas de acordo com as atividades presentes nesse processo.

3.3.2 Resultado da Pesquisa

O foco dessa entrevista foram as quatro atividades principais realizadas pelos sub-líderes e testadores: Cadastrar tarefas na ferramenta Jira, Criar cenário na ferramenta ALM, Desenvolver tarefas (*scripts* de teste) e Analisar *pull request* no GitHub[2].

[2] https://github.com/

A primeira pergunta realizada foi se existia alguma possível melhoria a ser feita na atividade “Cadastrar tarefas na ferramenta Jira” (entende-se por melhoria a adição de alguma atividade antes ou depois desta como também alguma mudança de comportamento/ações na própria atividade). Como resposta, os entrevistados sugeriram realizar uma reunião de planejamento e refinamento antes dessa atividade. Disseram que informalmente eles (sub-líderes) já fazem o refinamento (revisão) do documento antes de realizarem o cadastro das tarefas, mas que seria interessante fazer esse refinamento junto com a equipe. O refinamento trata-se de checar se as informações do documento estão atualizadas (se as telas apresentadas no documento e ações a serem realizadas condizem com a versão atual do sistema, por exemplo) e atualizá-las quando necessário. Além dessa revisão do documento, comentaram que seria interessante o estudo dele em conjunto com a equipe - entender o cenário de teste completo e elucidar possíveis dúvidas antes de começarem a desenvolver os *scripts* de teste.

Além disso, citaram a necessidade de realizarem o *planning poker* (BRASILEIRO, 2023) para estimar a complexidade de cada tarefa de desenvolvimento de *script* de teste, o que não acontece atualmente em conjunto com a equipe; essa estimativa é feita pelo sub-líder sozinho quando ele planeja os *cards* com as tarefas. Assim, a reunião sugerida pelos sub-líderes envolveria, então, três pontos: refinamento do documento, estudo do documento e estimativa de complexidade para desenvolvimento dos *scripts* de teste, sendo os três discutidos em conjunto com a equipe. Com essa atividade, todos os membros teriam um melhor entendimento do cenário a ser desenvolvido e também uma estimativa mais acurada da complexidade e do tempo necessário para terminá-lo.

A segunda pergunta envolvia a atividade “Criar cenário na ferramenta ALM”. Neste caso, foi comentada apenas uma ação que já estava sendo aplicada informalmente: a padronização de nomes dos cenários de teste. Para melhorar a organização e até o entendimento de cada cenário, seus nomes foram alterados seguindo um padrão de nomenclatura que remete ao tipo de ambiente e cliente que estão sendo utilizados em cada caso. Dessa forma, ao ler o nome do cenário, suas informações principais ficam evidentes (se o cenário é realizado no ambiente do cliente A ou B, por exemplo).

Ao perguntar sobre a atividade “Desenvolver tarefas (*scripts* de teste)”, o que foi sugerido pelos sub-líderes foi um estudo do documento do cenário em conjunto com a equipe antes de começar o desenvolvimento dos *scripts* de teste, já citado anteriormente.

Para a atividade “Analisar pull request no GitHub”, os entrevistados citaram uma ação que começou a ser aplicada há pouco tempo: *commits* diários no GitHub. O ato de sempre enviar o código mais atualizado para suas respectivas *branches* (ramificações do código principal do projeto - uma “cópia” do projeto onde cada desenvolvedor faz suas alterações no código sem sobrepor as alterações de outros desenvolvedores) não era um costume entre os membros da equipe, mas foi uma boa sugestão que acabou sendo adotada. A partir deste cenário, uma ideia de melhoria citada foi a automação do envio de *pull requests* no grupo da equipe. O *pull request* acontece quando o desenvolvedor acaba de desenvolver/alterar seu código e precisa integrá-lo ao código principal do projeto. Quando um *pull request* é solicitado é gerado um *link* e, atualmente, esse *link* é enviado em forma de comentário, manualmente, para o grupo da equipe de testes de regressão, mencionando o revisor do *pull request*, para que ele analise o código e faça o *merge* (aceite a inclusão do código novo ao código principal). A ideia seria automatizar essa ação de forma que, ao fazer um *pull request*, o comentário fosse enviado automaticamente para o grupo.

No final da entrevista, ao perguntar se existia mais algum ponto ou sugestão a ser adicionada, os entrevistados comentaram também que seria interessante realizar uma reunião no final da *sprint*. Essa reunião se faz necessária para que a equipe possa pontuar as ações que deram certo e o que precisa ser melhorado para o início da próxima *sprint*, por isso essa reunião é comumente chamada de reunião de retrospectiva.

3.3.3 Análise

A partir dessas informações foi possível entender as melhorias que podem ser aplicadas dentro do processo já existente de gerenciamento de mudanças planejadas, criando duas novas atividades e adicionando uma ação a uma atividade já existente.

A primeira ação será a criação de uma reunião de planejamento e refinamento antes da atividade “Cadastrar tarefas no Jira” - nesta reunião ocorre a revisão e estudo do documento do cenário, bem como o *planning poker* em conjunto com a equipe. Além disso, adicionar o envio automático do comentário do *pull request* como uma ação a ser feita dentro da atividade “Submeter código à ferramenta GitHub”. Por último, a criação de uma reunião ao final da *sprint*. Dessa forma, iremos propor um processo alinhado de acordo com diferentes visões e demandas dentro da equipe.

3.4 ANÁLISE GERAL

Após analisar as informações coletadas nas três entrevistas, foi possível entender como a equipe está organizada, como ocorre o gerenciamento de mudanças (não existem mudanças não planejadas - todas são planejadas, o problema é a comunicação da mudança para a equipe de teste) e as possíveis melhorias a serem implementadas no processo existente de mudanças planejadas. Ficou claro que quando a mudança é a adição de um novo cenário, o processo acontece no fluxo que foi apresentado na primeira entrevista (representado na Figura 1), sem demais problemas aparentes. Para este caso, foi criada uma tabela (Tabela 1) com as lacunas identificadas no processo de Gerenciamento de Mudanças Planejadas, que servirá de base para a proposição de melhorias.

Tabela 1 - Lacunas do cenário de Mudanças Planejadas

	Lacunas
1	Falta de estudo e revisão do documento do cenário novo junto à equipe
2	Falta da estimativa de complexidade para desenvolvimento *(planning poker)*
3	Falta da reunião ao final da *sprint*
4	Envio manual do link do *pull request*

Fonte: Elaborado pelo autor

Por outro lado, quando a mudança em questão está associada a alguma alteração em uma funcionalidade já existente, não há a comunicação formal desta mudança por parte da equipe de desenvolvimento para a equipe de teste de regressão. Como consequência, a equipe de teste poucas vezes toma conhecimento da mudança e, quando acontece, a informação chega sem detalhes importantes. Desta forma, os *scripts* de teste tendem a falhar quando executados durante o período de teste de regressão. Os *scripts* de teste replicam o comportamento que um usuário teria no software, desde digitar o e-mail e senha para efetuar login até clicar em botões para realizar uma compra, por exemplo. Assim, qualquer mudança feita pelos desenvolvedores nessas partes do sistema que já foram mapeadas para os *scripts* de teste pode fazer com que o *script* de teste falhe. A partir das lacunas identificadas no cenário de Alterações em Funcionalidades Existentes, foi criada uma tabela (Tabela 2) que também servirá de base para a proposição de melhorias.

Tabela 2 - Lacunas do cenário de Mudanças em funcionalidades já existentes

	Lacunas
1	A equipe não é informada adequadamente sobre as mudanças
2	Em casos em que a equipe de teste toma ciência da mudança, as informações sobre as mudanças não são detalhadas
3	Não atingimento de métricas definidas pelo cliente

Fonte: Elaborado pelo autor

O problema da falta de comunicação entre as equipes é que a equipe de teste leva mais tempo para executar os cenários que são impactados por estas mudanças durante o período de teste de regressão. A mudança em uma funcionalidade pode fazer com que o *script* de teste não funcione, caso esta mudança altere algum objeto na página (exclua um botão que já existia e estava mapeado no código, por exemplo), mude o fluxo do cenário (exclua ou adicione uma tela nova no sistema de carrinho de compras - uma tela para adicionar um novo serviço à sua compra, por exemplo), entre outros.

Como explicitado na seção 2.5 deste trabalho, sobre Gerenciamento de mudanças, caso o *script* de teste falhe, os membros da equipe de teste têm duas opções: executar o caso de teste manualmente (tomando cuidado para capturar as telas do sistema durante a execução, para guardar a evidência de que o caso de teste foi executado) e após acabar o teste de regressão corrigir o *script* de teste, ou parar para corrigir o *script* de teste durante o período do teste de regressão. Em ambos os casos, haverá impacto direto no trabalho do responsável pelo cenário de teste, o que pode impactar também no início da execução de outros cenários de teste e no processo de testes de regressão como um todo, levando a equipe ao *overworking*. Além disso, caso a execução do cenário seja manual, isso pode impactar em métricas que são monitoradas pelo cliente - como por exemplo, a métrica de automação. A métrica de automação mede a porcentagem de casos de teste que foram executados de forma automática, através dos *scripts* de teste. Quando um cenário é executado manualmente, a porcentagem dessa métrica diminui e pode ficar abaixo da meta pré-estabelecida, o que pode gerar pontos de atenção para o cliente.

4 PROPOSTAS DE MELHORIA

Após a análise dos dados coletados anteriormente, neste capítulo serão apresentadas as melhorias propostas para o processo Gerenciamento de Mudanças Planejadas e um processo para o Gerenciamento de Mudanças em Funcionalidades Existentes.

4.1 MELHORIAS NO PROCESSO DE MUDANÇAS PLANEJADAS

O Processo de Mudanças Planejadas atual da equipe possui algumas lacunas. Como apresentado no capítulo anterior, os principais pontos a serem melhorados são:

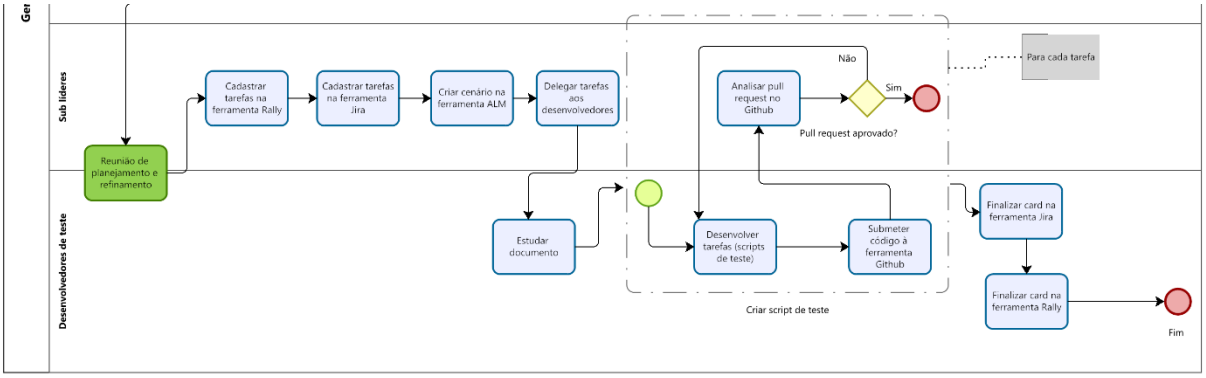
1) Falta de estudo e revisão do documento do cenário novo junto à equipe
2) Falta da estimativa de complexidade para o desenvolvimento *(planning poker)*
3) Falta da reunião ao final da *sprint*
4) Envio manual do link do *pull request*

Com o objetivo de preencher essas lacunas foram criadas atividades a serem inseridas no processo existente.

A primeira alteração foi a criação de uma atividade chamada “Reunião de planejamento e refinamento” logo após a liberação do documento pela equipe de estudos. Atualmente, após a liberação deste documento, os sub-líderes estudam o cenário novo sozinhos e já partem para a atividade de cadastramento das tarefas de desenvolvimento deste cenário nas ferramentas de controle. Essa reunião se faz necessária para que toda a equipe tenha conhecimento do cenário novo e para que possíveis dúvidas sejam sanadas antes de começarem a desenvolver os *scripts* de teste. Além disso, nessa reunião ocorreria o *planning poker*, que é a estimativa de complexidade que cada tarefa de desenvolvimento de *script* de teste possui; com essa estimativa de complexidade, é possível estimar também o tempo que a tarefa levará para ser concluída - essa técnica não é realizada atualmente.

A Figura 2 apresenta uma parte do processo de Gerenciamento de Mudanças Planejadas, mostrando na cor verde a nova atividade criada.

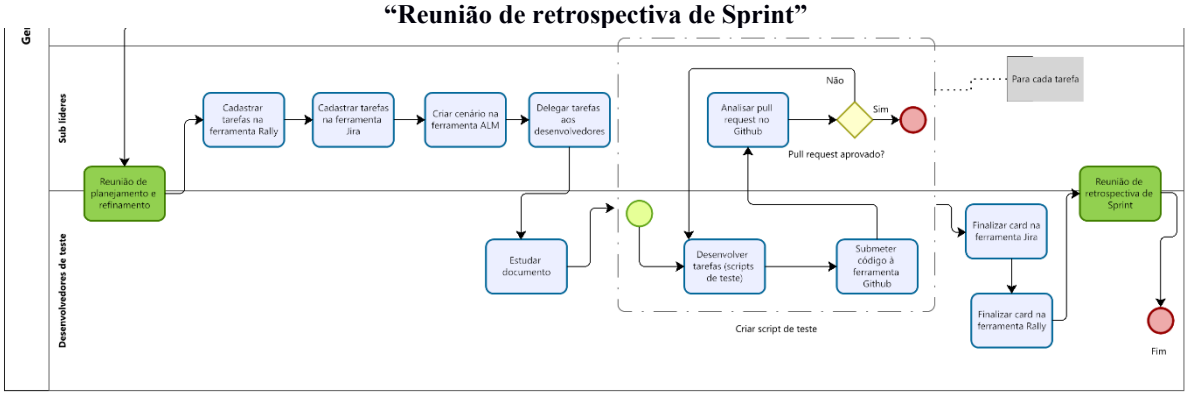
Figura 2 - Parte do Processo de Gerenciamento de Mudanças planejadas contendo a nova atividade "Reunião de planejamento e refinamento"



Fonte: Elaborado pelo autor

A segunda alteração no processo foi a criação da atividade "Reunião de retrospectiva da *sprint*" ao final do processo. A reunião de retrospectiva de *sprint* é uma prática do método Scrum bastante comum em equipes de desenvolvimento. O objetivo dessa atividade é analisar a *sprint* finalizada, mostrando os pontos fortes e fracos e sugerir possíveis melhorias para serem aplicadas na próxima *sprint*, visando sempre melhorar o processo de trabalho. Atualmente, essa atividade não existe na equipe e foi levantada pelos entrevistados como uma necessidade. Na Figura 3 está ilustrada uma parte do processo de Gerenciamento de Mudanças Planejadas, apresentando em verde a atividade nova inserida nele.

Figura 3 - Parte do Processo de Gerenciamento de Mudanças planejadas contendo a nova atividade "Reunião de retrospectiva de Sprint"



Fonte: Elaborado pelo autor

A terceira alteração no processo foi na comunicação de *pull requests*. Atualmente, como já explicitado no capítulo anterior, após fazer o *pull request* do código, a comunicação dele é feita através de um grupo com os membros da equipe, onde a pessoa que fez o *pull request* posta manualmente o seu link, mencionando o avaliador. Foi comentado por um dos membros que este processo de aviso de *pull request* a ser analisado pode ser automatizado. Desta forma,

foi incluído no processo o envio automático do link no grupo. Essa melhoria foi incluída dentro da atividade já existente de “Submeter código à ferramenta GitHub”.

A Tabela 3 apresenta a relação entre as lacunas identificadas no processo de mudanças planejadas (Tabela 1) e suas respectivas melhorias aplicadas, bem como os benefícios que estas melhorias trazem à equipe.

Tabela 3 - Relação das melhorias implementadas e seus respectivos benefícios

Número da Lacuna	Melhoria	Benefícios
1 e 2	Reunião de planejamento e refinamento	● Entendimento completo do cenário que será desenvolvido; ● Possíveis dúvidas sobre o cenário são discutidas e entram em conhecimento de toda equipe; ● Estimativa de complexidade de desenvolvimento dos *scripts* de teste feita com a participação da equipe.
3	Reunião de retrospectiva de sprint	● Alinhamento da equipe em relação às ações que deram certo durante a sprint e o que precisa ser melhorado para a próxima; ● Aumenta a produtividade e a relação entre os membros da equipe; ● Melhora a qualidade do que está sendo desenvolvido.
4	Envio automático do link do *pull request*	● Agiliza o processo de aviso ao revisor do *pull request*; ● Evita falhas humanas (como esquecer de enviar o link, por exemplo).

Fonte: Elaborado pelo autor

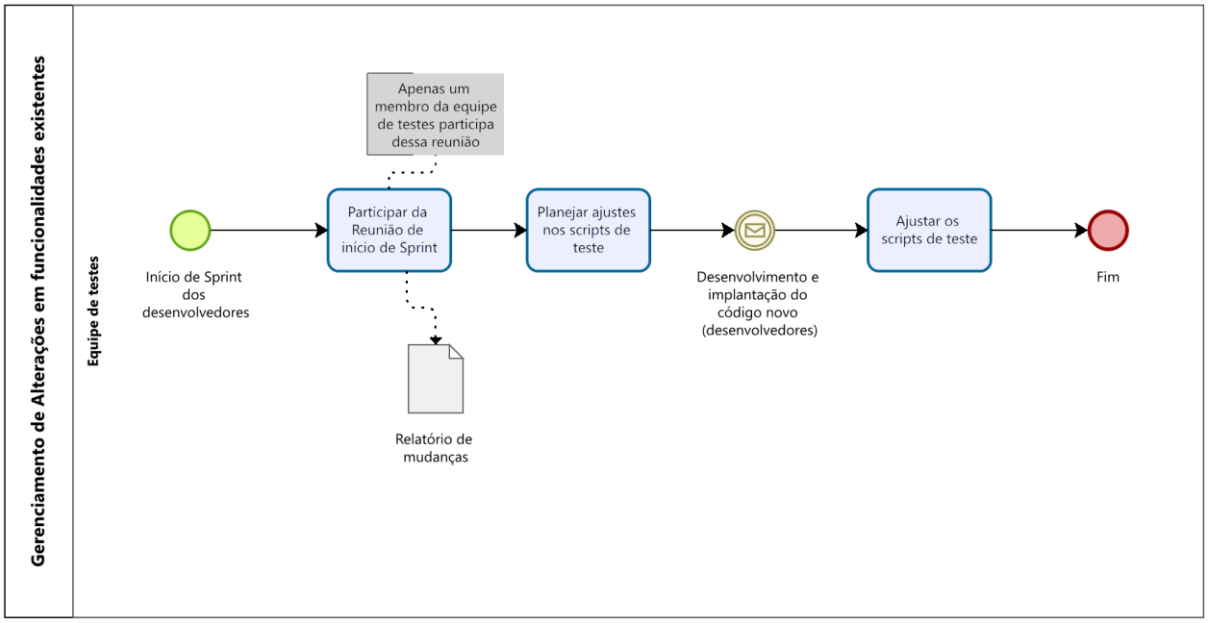
Com essas três alterações no processo, as lacunas de melhorias são preenchidas, o fluxo de trabalho torna-se mais completo e o desenvolvimento de *scripts* de teste tende a ser mais eficiente. O entendimento dos cenários a serem desenvolvidos é adquirido logo na Reunião de planejamento e refinamento, facilitando o trabalho dos desenvolvedores dos *scripts* de teste. A melhoria das relações entre os membros e das ações tomadas por eles no dia a dia também

acontecem com a realização da Reunião de retrospectiva de *sprint*, necessária para analisar o andamento do time e suas entregas. Com a automação do envio de *pull requests*, os desenvolvedores ficam com menos uma tarefa para realizar, otimizando o tempo de trabalho.

4.2 PROCESSO DE GERENCIAMENTO DE ALTERAÇÕES EM FUNCIONALIDADES EXISTENTES

Além das melhorias propostas para o Processo de Mudanças planejadas, foi criado um processo para organizar o fluxo de trabalho em casos de alterações em funcionalidades já existentes. Foi levantado durante as entrevistas que não existe um canal direto de comunicação entre os desenvolvedores e a equipe de testes para repassar a informação das mudanças em funcionalidades existentes - são equipes que pertencem a empresas diferentes. Assim, tendo como base as informações coletadas com os entrevistados, foi elaborado um processo para este caso específico de mudanças (Figura 4).

Figura 4 - Processo de Gerenciamento de Alterações em Funcionalidades Existentes



Fonte: Elaborado pelo autor

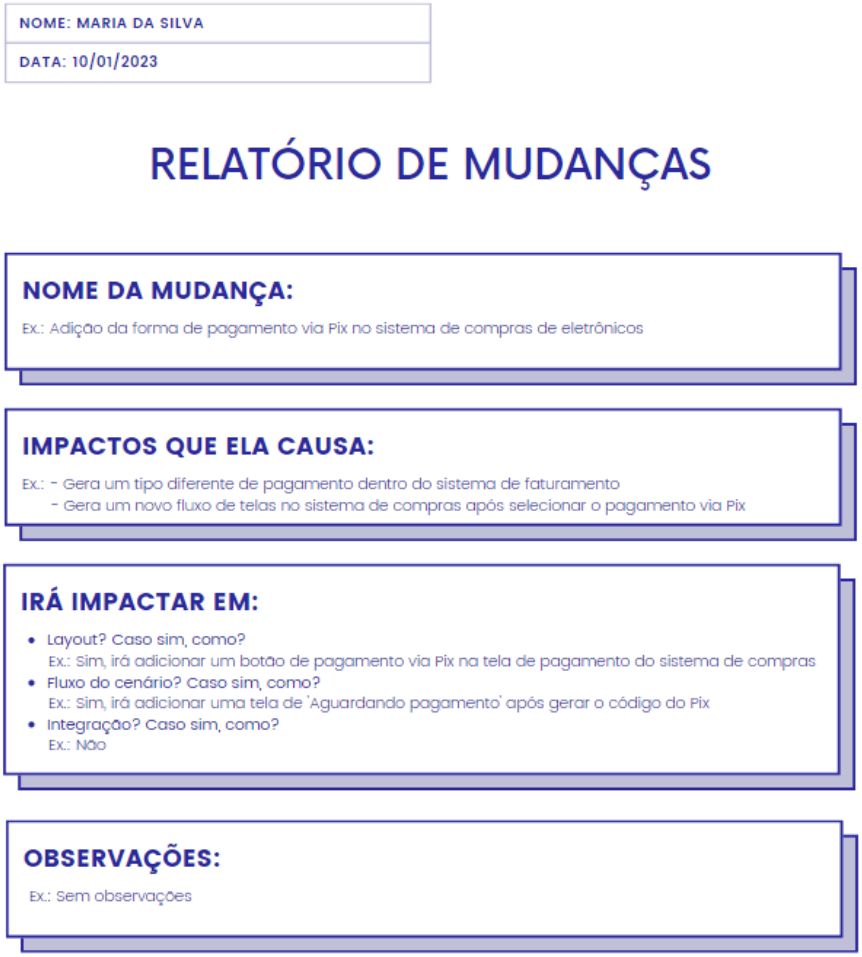
O ponto de partida deste processo é o início da *sprint* da equipe de desenvolvedores. A primeira atividade do processo criado é “Participar da reunião de início de *sprint*”, criada com o intuito de mapear as mudanças que serão implementadas no sistema e se elas irão impactar na equipe de testes de regressão. Apenas um membro da equipe de testes estará presente nesta reunião e seu papel principal será realizar perguntas e obter informações sobre as alterações apresentadas, tomando um cuidado especial para avaliar se elas irão impactar em três pontos

importantes que afetam diretamente a equipe de testes de regressão: fluxo do cenário, layout da página e integração entre sistemas. Caso as alterações impactem em pelo menos um destes pontos, será necessário o ajuste dos *scripts* de teste. Para ilustrar como esses três pontos afetam os testes de regressão, serão apresentados três cenários:

- **Fluxo do cenário:** imaginando um site de compra de produtos eletrônicos no qual será adicionada uma nova página onde será possível escolher se a sua compra será entregue embrulhada em papel de presente e com uma mensagem escrita. Essa nova tela precisaria ser adicionada no *script* de teste já existente. Para melhor visualização, podemos dizer que o *script* de teste existente realiza os seguintes passos: 1-Cadastrar informações do cliente, 2-Selecionar produto eletrônico, 3-Realizar pagamento; enquanto o sistema, após a adição desta nova tela, possui agora o seguinte caminho: 1-Cadastrar informações do cliente, 2-Selecionar produto eletrônico, 3-Escolher forma de entrega do produto, 4- Realizar pagamento. Caso o *script* de teste execute os passos neste novo caminho, sem realizar nenhum ajuste, irá falhar, pois não reflete o fluxo atual implementado no sistema.
- **Layout da página:** utilizando o exemplo do sistema de compra de produtos eletrônicos citado anteriormente, suponha que na página de realizar o pagamento da compra exista um botão chamado ‘Concluir compra’ localizado do lado direito da tela. Caso este botão seja mudado para o lado esquerdo, ou tenha seu nome alterado para ‘Finalizar compra’, é possível que o *script* de teste falhe. Qualquer mudança, por mais simples que seja, pode fazer com que o código do *script* de teste precise ser atualizado, porque os elementos da UI (Interface do Usuário) são diretamente referenciados nos *scripts* de teste para possibilitar a automação.
- **Integração entre sistemas**: um exemplo de impacto na integração entre sistemas seria a mudança do sistema que armazena os pedidos de compra após concluídos. Utilizando o mesmo cenário anterior, após finalizar a compra, é possível consultar os detalhes do pedido feito e as informações do cliente em um outro sistema de controle utilizado pelos administradores. Caso este sistema de controle A mude para um novo sistema B, ocorrerá um impacto na integração. Os pedidos de compras feitos após esta mudança estarão disponíveis agora no sistema B, e não mais no sistema A. Assim, os *scripts* precisarão ser atualizados para refletir este novo sistema e suas especificações.

Para facilitar o registro das mudanças durante a reunião, o membro da equipe de testes deverá preencher um documento chamado de “Relatório de mudanças”, que possui perguntas específicas a serem respondidas. Na Figura 5 encontra-se o *template* do Relatório de Mudanças com exemplos de informações fictícias preenchidas.

Figura 5 - Template do Relatório de Mudanças



Fonte: Elaborado pelo autor

A próxima atividade da equipe de testes é “Planejar ajustes nos *scripts* de teste”, na qual o “Relatório de mudanças” será apresentado à equipe de testes para que todos tomem ciência das mudanças que serão implementadas e para definir como a equipe se organizará para corrigir o que for necessário nos *scripts* de teste.

Para finalizar o processo, a equipe de testes de regressão aguarda até que a equipe de desenvolvimento disponibilize a nova versão do sistema para que, finalmente seja possível realizar os ajustes nos *scripts* de teste de acordo com a nova versão do sistema.

4.3 AVALIAÇÃO DAS ALTERAÇÕES PROPOSTAS

A avaliação das melhorias propostas no processo de Gerenciamento de Mudanças Planejadas e do Processo de Gerenciamento de Alterações em funcionalidades existentes foi inspirada na metodologia TAM (*Technology Acceptance Model* - Modelo de Aceitação de Tecnologia) (DAVIS, 1989), onde buscou-se avaliar a utilidade e viabilidade dessas mudanças.

Para isso, foram realizadas duas entrevistas de *feedback* com os membros da equipe que participaram das entrevistas anteriores. A primeira entrevista foi realizada com os sub-líderes e a segunda com a arquiteta de teste. Ambas aconteceram por videochamada e tiveram uma duração de aproximadamente 20 minutos cada. Não foi possível realizar uma entrevista com o líder de automação pois, quando estas reuniões foram realizadas, ele não fazia mais parte da equipe. As entrevistas realizadas procuraram entender se as melhorias propostas e o processo criado facilitariam o trabalho da equipe em seu cotidiano (perspectiva da utilidade) e se eram possíveis de serem implementadas (perspectiva da viabilidade), bem como coletar sugestões do que poderia ser mudado nos processos propostos.

4.4 RESULTADOS DA AVALIAÇÃO

Durante as entrevistas de *feedback*, o foco das primeiras perguntas esteve nas três atividades/ações criadas dentro do processo de Gerenciamento de Mudanças Planejadas: ‘Reunião de planejamento e refinamento’, ‘Envio automático de *pull request*’ e ‘Reunião de retrospectiva de *sprint’.* As entrevistas foram conduzidas citando cada atividade e realizando as seguintes perguntas:

1. Essa mudança irá melhorar/facilitar o trabalho da equipe? Por quê?

2. Essa mudança é viável?

Ao perguntar para os sub-líderes sobre a atividade ‘Reunião de planejamento e refinamento’, eles disseram que essa atividade facilitaria o trabalho por esclarecer as possíveis dúvidas do cenário de teste e melhorar o conhecimento do cenário como um todo, por ele estar sendo estudado em conjunto com a equipe antes do início do desenvolvimento. Durante a entrevista com a arquiteta de teste, ela comentou que o melhor entendimento do cenário acaba minimizando o desenvolvimento errado dos *scripts* de teste. Os sub-líderes também comentaram sobre o *planning poker*, que está incluído dentro dessa reunião. Eles disseram que pelo fato de a estimativa de complexidade também estar sendo feita no início, antes do desenvolvimento, e com toda a equipe, essa métrica de estimativa se torna mais real. Sobre a

viabilidade ou não desta reunião, os entrevistados disseram que é possível a realização dela sem problemas, necessitando apenas de uma organização interna da equipe.

O segundo ponto a ser apresentado foi o 'Envio automático de *pull request*', ação que foi incluída dentro da atividade 'Submeter código à ferramenta GitHub'. Em relação à esta ação, os sub-líderes comentaram que ela melhoraria o trabalho do dia a dia por facilitar o controle dos envios de código. Ao questionar sobre a viabilidade desta ação, foi dito pelos sub-líderes que é possível ser implementada, mas depende de realizar configurações adicionais no ambiente. Existe uma forma de vincular a ferramenta GitHub à ferramenta de comunicação (mensagens) utilizada pela equipe, mas essa configuração requer autorização prévia da empresa. Dessa forma, precisaria também do entendimento de como funciona este processo de autorização. Durante a entrevista com a arquiteta de teste, ela comentou que é possível conversar com colegas de outros projetos que utilizam as mesmas ferramentas para entender melhor como elas funcionam e entender o processo de autorização.

O terceiro ponto comentado foi sobre a atividade 'Reunião de retrospectiva de *sprint*'. Os sub-líderes disseram que esta atividade melhoraria o trabalho por levantar os pontos que deram certo e os que deram errado durante a sprint, e que de acordo com os pontos que deram errado seriam criadas ações para serem realizadas na próxima *sprint*. Sobre a viabilidade desta reunião, também comentaram que é possível e requer apenas organização interna.

Após as perguntas sobre as melhorias propostas no modelo de Gerenciamento de Mudanças Planejadas, foram realizadas perguntas sobre o modelo de Gerenciamento de Alterações em funcionalidades existentes. Neste, foi apresentado o modelo do novo processo criado (Figura 5), estando em foco todas as atividades presentes nele. Também foi perguntado se essas atividades melhorariam/facilitariam o trabalho e se eram viáveis de ser implementadas.

Ao perguntar sobre o processo em geral, os sub-líderes disseram que a atividade 'Participar da reunião de início de *sprint* dos desenvolvedores' (atividade principal do processo) facilitaria bastante o trabalho pelo fato da equipe de teste poder se planejar para aplicar as correções nos *scripts* com antecedência, após obterem as informações necessárias coletadas nessa reunião. Comentaram também que, com as devidas correções aplicadas nos *scripts*, a quantidade de defeitos abertos durante o período do teste de regressão diminuiria, pois boa parte dos defeitos são abertos quando o sistema apresenta alguma mudança em seu *layout* ou no fluxo do cenário de teste que não foi comunicada previamente (um defeito é aberto, normalmente, quando o sistema apresenta algum erro ou não se comporta da forma esperada durante a

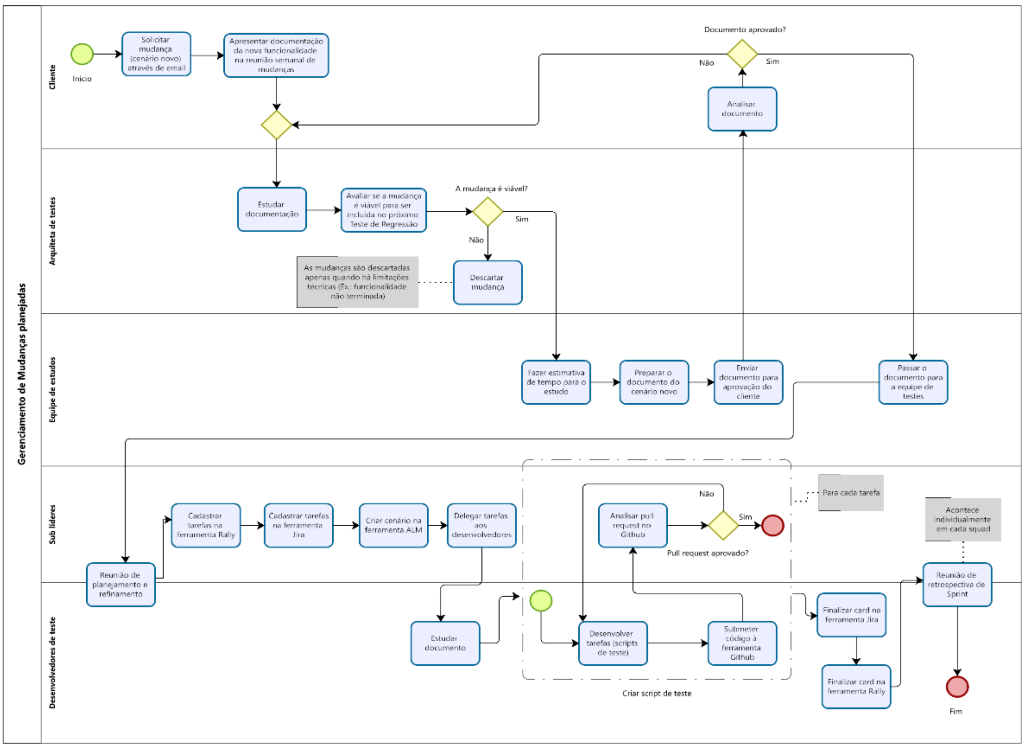
execução dos *scripts* de teste). Em relação à viabilidade, foi comentado pela arquiteta de teste que é possível essa participação na reunião dos desenvolvedores acontecer, mas requer ajustes internos por envolver questões contratuais - a equipe de teste e a equipe de desenvolvimento pertencem a empresas diferentes.

4.5 MELHORIAS NOS PROCESSOS

Durante as entrevistas apresentadas anteriormente foram levantados alguns pontos que podem ser melhorados dentro das atividades e do processo que foram apresentados.

O primeiro ponto comentado por um dos sub-líderes foi em relação à atividade 'Realizar reunião de retrospectiva de *sprint*' do processo de Gerenciamento de Mudanças Planejadas. Quando essa atividade foi adicionada ao processo, ela foi pensada para ser realizada em conjunto com todos os membros da equipe de teste. Apesar disso, foi sugerido por um dos sub-líderes que esta atividade seja realizada internamente dentro de cada *squad*. A justificativa para ela ser realizada dessa forma foi que cada uma das 5 *squads* possui seus sistemas específicos e suas particularidades, então caso essa reunião fosse realizada com toda a equipe, poderia ficar desorganizada e não ser eficiente. Dessa forma, foi adicionado um comentário sobre essa reunião no processo de Gerenciamento de Mudanças Planejadas (Figura 6).

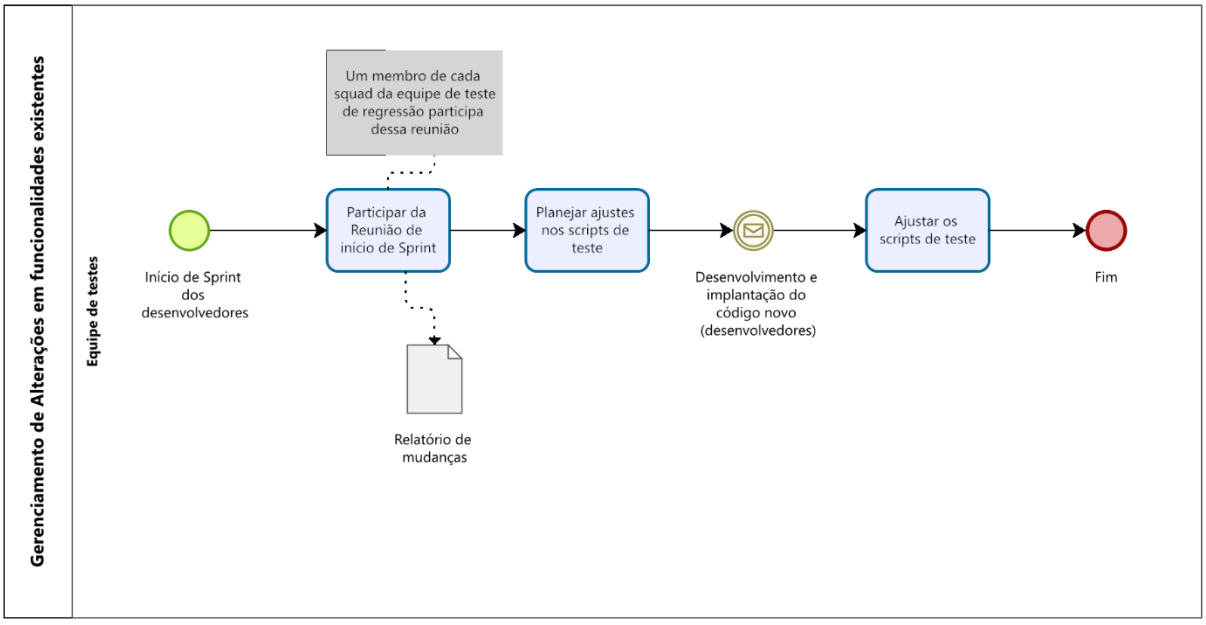
Figura 6 - Processo de Gerenciamento de Mudanças Planejadas Final



Fonte: Elaborado pelo autor

Outro ponto levantado foi pela arquiteta de teste, em relação à atividade ‘Participar da reunião de início de *sprint* dos desenvolvedores’, do processo de Gerenciamento de Mudanças em Funcionalidades existentes. Ao elaborar o processo, essa atividade foi pensada tendo apenas um membro da equipe de teste participando da reunião. Em contrapartida, a arquiteta de teste comentou que seria necessário ter um membro de cada *squad* nessa reunião. A justificativa dela é similar à justificativa anterior - cada *squad* possui sistemas diferentes e suas especificidades; um membro da *squad mobile* não entenderia nem saberia perguntar sobre uma informação específica do sistema de faturamento, por exemplo. Dessa forma, faz-se necessário um membro de cada *squad* participar dessa reunião, visto que os detalhes sobre as mudanças são específicos e requerem conhecimento técnico de cada sistema, seja para fazer anotações sobre a mudança ou para tirar dúvidas com os desenvolvedores. Além disso, a arquiteta comentou que seria necessário que estes membros de cada *squad* tivessem pelo menos 1 ano como funcionário efetivo, pois este trabalho requer conhecimentos e habilidades que um estagiário ou recém-contratado não possuem. A partir desta avaliação, esse ajuste foi implementado no processo de Gerenciamento de Alterações em Funcionalidades Existentes (Figura 7).

Figura 7 - Processo de Gerenciamento de Alterações em Funcionalidades Existentes Final



Fonte: Elaborado pelo autor

O último ponto levantado pelos sub-líderes não está relacionado a nenhuma atividade específica dos processos, mas sim à organização interna da equipe. Eles comentaram que acham interessante separar melhor as funções dentro das *squads*. Como foi apresentado no Capítulo 3 deste trabalho, os membros da equipe geralmente possuem duas atribuições: desenvolver os

scripts de teste e executá-los durante o teste de regressão. Os sub-líderes disseram que seria bom separar, dentro de cada *squad*, os membros que irão apenas desenvolver os *scripts* e os membros que irão apenas executá-los. Dessa forma, durante o período do teste de regressão, o desenvolvimento de novos *scripts* continua ocorrendo. Atualmente, todos os membros param o desenvolvimento dos *scripts* de teste da *sprint* em que estão para executar o teste de regressão, o que acaba atrasando a entrega de novos cenários de teste. Com a reestruturação das *squads*, o trabalho seria mais eficiente e a gerência conseguiria enxergar com mais clareza a necessidade de alocar melhor as pessoas.

4.6 LIMITAÇÕES DA AVALIAÇÃO

Como apresentado anteriormente, as entrevistas de *feedback* foram realizadas com a arquiteta de teste e os sub-líderes, os mesmos membros que participaram das primeiras entrevistas. Não foi possível ter o *feedback* do líder de automação pois ele não fazia mais parte da equipe quando essas reuniões foram realizadas. Em uma situação ideal, a avaliação deveria ter contado com outros integrantes, além dos que participaram das entrevistas iniciais, para que o viés no seu resultado fosse minimizado. Entretanto, a escolha destes membros se deu pelo fato deles serem gestores de equipe e terem uma visão mais abrangente das necessidades da equipe como um todo, podendo contribuir de forma mais efetiva com os pontos que poderiam ser melhorados.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

O objetivo inicial deste trabalho era entender como estava organizado o processo de gerenciamento de mudanças dentro de uma equipe de teste de regressão e propor melhorias a ele. Dessa forma, para entender o processo foram organizadas entrevistas com membros da equipe de testes de regressão.

A crença inicial era de que existia apenas um processo de mudanças, mas após a primeira entrevista foi possível entender que existiam dois tipos de mudanças diferentes, que requeriam ações diferentes. Um processo já estava estruturado e já era seguido dentro da equipe, enquanto o outro era inexistente.

A segunda entrevista foi importante para entender que o problema estava na má comunicação entre as equipes quando a mudança ocorria em funcionalidades já existentes. Com as informações dadas pela entrevistada, foi possível estruturar um processo para ser seguido nesta situação. Já na terceira entrevista, foi possível ouvir a opinião dos sub-líderes sobre melhorias para implementar no Processo de Gerenciamento de Mudanças Planejadas.

A investigação por meio de entrevistas foi necessária pelo fato de não existir documentação explicando o processo de gerenciamento de mudanças que é seguido dentro da equipe. Após a coleta e análise das informações, foi criado um desenho do processo (Figura 1), o que tornou o entendimento dele mais fácil. A partir deste desenho, foi possível propor melhorias dentro do processo.

Durante todas as entrevistas (incluindo as de *feedback*) os membros entrevistados foram bastante solícitos e participativos, mostrando interesse pela pesquisa. Foi notado pelos membros da equipe que as melhorias apresentadas eram contribuições relevantes para o seu dia a dia de trabalho.

Assim, o objetivo principal do trabalho foi alcançado: foram propostas melhorias e suas viabilidades e utilidades foram analisadas. O processo que existia apenas de forma tácita agora está documentado, além de ter sido criado um processo para uma situação específica que não estava organizada.

5.2 TRABALHOS FUTUROS

Como forma de dar continuidade a este trabalho, seria interessante aplicar de fato estas melhorias nos processos e analisar se o trabalho se tornou realmente mais eficiente. Com essa nova análise, seria possível aperfeiçoar o processo de acordo com possíveis novas demandas dentro da equipe.

Outro ponto a ser explorado seria realizar entrevistas com outros membros da equipe para extrair diferentes visões e necessidades de melhoria dentro dos processos, visto que as entrevistas foram realizadas com um escopo reduzido de pessoas, pertencentes à equipe de testes de regressão.

Além das entrevistas com outros membros da equipe, o trabalho poderia ter continuidade analisando a equipe de desenvolvimento. Realizar pesquisas com os membros para entender como ela funciona e os processos que segue, visando o aperfeiçoamento dos processos internos da equipe, alinhando-os aos processos da equipe de teste de regressão.

Por fim, outra forma de dar continuidade a este trabalho seria descrever as atividades que estão dentro dos processos de Gerenciamento de Mudanças Planejadas e o processo de Gerenciamento de Alterações em Funcionalidades Existentes. A descrição das tarefas auxiliaria no entendimento dos processos e poderia também servir de documentação para ser utilizada dentro da própria equipe de teste de regressão.

1. REFERÊNCIAS BIBLIOGRÁFICAS

AMMANN, P.; OFFUTT, J. *Introduction to Software Testing*. New York: Sheridan Books, 2017. p. 406-409

BRASILEIRO, R. *Planning Poker: A melhor maneira de estimar qualquer atividade*. Disponível em: <https://www.metodoagil.com/planning-poker/>. Acesso em: 01 de fev. de 2023

BOURQUE, P.; FAIRLEY, R. E. *SWEBOK Guide to the Software Engineering Body of Knowledge.* v.3. Washington: IEEE Computer Society, 2014. p. 84-87

DAVIS, F. D. *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. MIS Quarterly, 13(3), 319–340, 1989.

International Organization for Standardization. ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

OMG. BUSINESS PROCESS MODEL & NOTATION (BPMN), c2023. Disponível em: <https://www.omg.org/bpmn>. Acesso em: 12 de jan. de 2023.

PEZZE, M.; YOUNG, M. *Software Testing and Analysis: Process, Principles, and Techniques*. New Jersey: John Wiley & Sons, Inc., 2008. p. 449

PRESSMAN, R. S.; MAXIM, B. R. *Software Engineering: A Practitioner's Approach*. 9. ed. New York: McGraw-Hill Education, 2020. p. 312-347

SOMMERVILLE, Ian. *Engenharia de software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

2. APÊNDICE A - Questionário da Pesquisa Preliminar

1) Identificar a estrutura da equipe:

- Como a equipe está dividida?
- Quantas pessoas?
- Quais os papéis desempenhados?

2) Identificar as entradas (como a comunicação da mudança chega à equipe):

- Existe um documento específico que descreve a mudança? Qual?
    - Se não, de que formas essa comunicação costuma chegar?
- Pode ser enviada por e-mail, telefone etc., ou existe um canal definido para isso?
- Quem recebe essa comunicação (existe um ponto focal para esse recebimento)?
- O que essa pessoa faz quando essa comunicação chega?
- Existe algum tipo de classificação ou categorização da mudança?
- É feito algum tipo de estimativa (especialmente de prazo)? Quais? Quem são os responsáveis?

3) Identificar como a equipe de testes trata essa comunicação de mudança:

- Quando a equipe de testes é envolvida, após a comunicação da mudança?
- Como a equipe de testes de regressão recebe a comunicação sobre a mudança?
- Como a equipe de testes de regressão atua a partir desse momento?
- Tarefas que ela deve executar.

4) Identificar o que a equipe de testes deve entregar (saída):

- Quais as saídas esperadas da equipe de testes, ou seja, que entregáveis ela deve gerar nesse processo?

Importante: ao final da entrevista, verificar se o entrevistado indica outra pessoa da equipe que pode dar mais detalhes sobre alguns desses pontos.