GIOVANNI VICENTE GENTILE DA SILVA

Utilização de *Behaviour-Driven Development* para apoio à Gestão das Comunicações

São Paulo

2021

GIOVANNI VICENTE GENTILE DA SILVA

Utilização de *Behaviour-Driven Development* para apoio à Gestão das Comunicações

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Profª. Drª. Jussara Pimenta Matos

São Paulo

2021

[verso da folha de rosto]

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

FICHA CATALOGRÁFICA

[Colocar na versão final do trabalho. Obter em: <https://www.poli.usp.br/bibliotecas/servicos/catalogacao-na-publicacao>]

Nome: [SOBRENOME, Nome do aluno]

Título: [Título da monografia]

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em Tecnologia de Software.

Aprovado em:

Banca Examinadora

Prof(a). Dr(a). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instituição: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Julgamento: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof(a). Dr(a). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instituição: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Julgamento: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof(a). Dr(a). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Instituição: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Julgamento: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DEDICATÓRIA

*Em memória de meus avós Giovanni e Isabella.*

AGRADECIMENTOS

À Universidade de São Paulo – USP que [bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla].

À Escola Politécnica da Universidade de São Paulo – EPUSP que [bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla].

Ao PECE – Programa de Educação Continuada em Engenharia que [bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla]

Aos meus pais que [bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla].

RESUMO

[SOBRENOME, Nome abreviado]. [Título da monografia]. [Ano]. [Número de folhas]. Monografia (MBA em Tecnologia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. [Ano].

[Deve conter motivações, objetivos, métodos empregados, resultados e conclusões em 1 parágrafo]

Palavras-chave: [bla, bla, bla, bla, bla]

ABSTRACT

[SOBRENOME, Nome abreviado]. [Título da monografia]. [Ano]. [Número de folhas]. Monografia (MBA em Tecnologia de Software). Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo. São Paulo. [Ano].

[Should have motivations, objectives, applied methods, results and conclusions in 1 paragraph]

Keywords: [bla, bla, bla, bla, bla]

LISTA DE ILUSTRAÇÕES

Figura 1 – Título da figura .................................................................……………… xx

Figura 2 – Título da figura .................................................................……………… xx

Figura 3 – Título da figura .................................................................……………… xx

LISTA DE TABELAS

Tabela 1 – Título da tabela ..................................................................................... xx

Tabela 2 – Título da tabela ..................................................................................... xx

Tabela 3 – Título da tabela ..................................................................................... xx

LISTA DE ABREVIATURAS E SIGLAS

[BLA Bla bla bla bla bla bla bla bla bla bla bla bla bla]

[BLA Bla bla bla bla bla bla bla bla bla bla bla bla bla]

SUMÁRIO

1. INTRODUÇÃO ............................................................................................... xx

1.1 Motivações........................................................................................................ xx

1.2 Objetivo ............................................................................................................ xx

1.3 Justificativas .................................................................................................... xx

1.4 Método de Pesquisa......................................................................................... xx

1.5 Estrutura do Trabalho ....................................................................................... xx

2. REVISÃO BIBLIOGRÁFICA... .......................................................................... xx

2.x Título da Seção................................................................................................. xx

2.y Considerações do Capítulo .............................................................................. xx

3. DESENVOLVIMENTO………. ........................................................................... xx

3.x Título da Seção………………. ......................................................................... xx

3.y Considerações do Capítulo ............................................................................. xx

4 ANÁLISE DE RESULTADOS ............................................................................ xx

4.x Título da Seção………………. ......................................................................... xx

4.y Considerações do Capítulo .............................................................................. xx

5. CONSIDERAÇÕES FINAIS................................................................................ xx

5.1 Conclusões....................................................................................................... xx

5.2 Contribuições do Trabalho .............................................................................. xx

5.3 Trabalhos Futuros ........................................................................................... xx

REFERÊNCIAS ..................................................................................................... xx

GLOSSÁRIO ......................................................................................................... xx

APÊNDICE A – [Título do Apêndice]............. ........................................................ xx

ANEXO 1 – [Título do Anexo] .............................................................................. xx

1. INTRODUÇÃO

Este capítulo apresenta as motivações, objetivo, justificativas, método de pesquisa e a estrutura do trabalho.

* 1. Motivações

No contexto de transformação digital que o mercado de tecnologia está vivenciando, a busca pela excelência na entrega dos *softwares* com qualidade, ou seja, relizando-a dentro do prazo e custo, além de gerar valor dentro das expectativas do negócio, com centralidade no cliente e adaptabilidade a mudanças se faz cada vez mais necessária, e é preciso adotar técnicas e práticas para atingimento destes objetivos.

Dentre os fatores que podem influenciar de forma desfavorável os resultados de um projeto e o atingimento dos objetivos citados no parágrafo anterior, estão problemas no entendimento dos requisitos e ruídos nas comunicações (VERNER; SAMPSON; CERPA, 2008), principalmente em projetos que possuem atuação de várias equipes distribuídas (HINDS; BAILEY, 2003).

Diante deste cenário, há a necessidade de proposição e aplicação de medidas para minimizar os problemas causados por conta destes fatores, validando-as em um projeto real de desenvolvimento de *software*, em um contexto no qual as equipes de desenvolvimento são impactadas por estes problemas.

Duas práticas podem ser aplicadas para mitigar estes impactos: o BDD (*Behaviour-Driven Development*; em português, Desenvolvimento Dirigido a Comportamentos) para especificação de requisitos e a Gestão das Comunicações conforme proposto pelo PMBOK®.

*Behaviour-Driven Development* consiste em um conjunto de práticas, tipicamente aplicadas em um contexto de desenvolvimento ágil, pois foi oriundo do *Test-Driven Development* que é um conceito introduzido pelos métodos ágeis (OLIVEIRA, 2015), que procura criar uma linguagem ubíqua através de uma sintaxe conhecida como *Gherkin* (WYNNE; HELLESØY, 2017), ou seja, todos os envolvidos pronunciam, referenciam e têm interpretação idêntica dos conceitos tratados, e facilitar o entendimento de quais critérios precisam ser satisfeitos para atingimento dos objetivos do projeto, visto que busca descrever o comportamento esperado do *software* (NORTH, 2006) aumentando a eficiência das comunicações, principalmente entre os times técnicos e times que representam o negócio, bem como a probabilidade de sucesso dos projetos (HÄSER *et al*., 2016).

O PMBOK® é um guia de boas práticas para gerenciamento de projetos, dentre as disciplinas contidas no guia está a Gestão de Comunicações, que visa garantir a geração, coleta, armazenamento, recuperação e destinação das informações geradas ao longo do projeto, de modo a procurar minimizar problemas relacionados à quaisquer ruídos nestas comunicações.

Comunicação não abrange somente a transmissão de uma mensagem, mas também a compreensão do significado desta mensagem entre os envolvidos, portanto o BDD pode se tornar um importante aliado à Gestão das Comunicações, principalmente para definição de requisitos e de seus critérios de aceite, de modo a contribuir com o andamento e entrega do projeto e atacar os problemas citados ao longo desta seção.

* 1. Objetivo

Propor a utilização das práticas de BDD para definir o comportamento e critérios de aceite do *software* (como abordagem de Engenharia de Requisitos), em conjunto com as definições de Gestão das Comunicações do PMBOK®, para avaliar a influência destas abordagens na diminuição de ruídos nas comunicações e compreensão dos requisitos bem como o seu auxílio na finalização bem-sucedida de um projeto com envolvimento de múltiplas equipes.

Estas propostas serão definidas em uma lista de práticas que buscam alinhar as definições dos comportamentos do sistema em uma linguagem ubíqua com o controle das comunicações do projeto, de modo a assegurar que as informações e definições sejam transmitidas de forma clara e atualizada para as partes interessadas.

Cada item desta lista deve ser cumprido ao longo da realização do trabalho, para posterior avaliação da relevância na utilização destas práticas no resultado final.

* 1. Justificativas

De acordo com o estudo de (VERNER; SAMPSON; CERPA, 2008), que avaliou o andamento de oito (8) projetos de desenvolvimento de *software* que não obtiveram sucesso, decisões tomadas sem o entendimento pleno dos requisitos do sistema estiveram entre os fatores responsáveis pela falha de 73% dos projetos avaliados.

Um artigo publicado pelo PMI relata que mais da metade do orçamento de um projeto entra em risco logo em seu início devido à comunicações ineficientes, o que mostra que estes ruídos acarretam em maiores custos e podem fazer com que um projeto falhe ou seja entregue fora do orçamento e até mesmo prazo (PMI, 2013).

Estes problemas não são novos em desenvolvimento de software, inclusive foram citados por Frederick Brooks em seu livro *The Mythical Man-Month*, que possui um capítulo destacando que adicionar mais pessoas a um projeto atrasado tende a atrasá-lo ainda mais, pois dentre outros fatores, adicionam novos canais de comunicação que se não forem gerenciados podem implicar em mais problemas do que soluções (BROOKS, 1975).

Com o advento das metodologias ágeis, a importância de realizar as comunicações de forma efetiva foi muito enfatizada e gerou uma vasta literatura (MAZUCA, 2018), porém estudos como os citados nos parágrafos anteriores corroboram que ainda temos oportunidades de melhoria e a discussão deste assunto segue relevante.

Outro fator que reforça a necessidade de explorarmos estes assuntos, é o de que em projetos de grande porte, há o envolvimento de diferentes times, que por vezes estão distribuídos, e consequentemente existem problemas de comunicação por conta desta homogeneidade que faz com que existam muitos canais de comunicação para gerenciar (HINDS; BAILEY, 2003).

Estes pontos também contribuem para uma maior incidência de conflitos, tipicamente causados por expectativas desalinhadas entre os times envolvidos em projetos nestes contextos (WILLIAMS; O’REILLY, 1998), evidenciando também oportunidades de melhorias na modelagem dos requisitos e no alinhamento sobre eles.

Devemos levar em consideração que não adianta mapear todos os canais de comunicação e atualizá-los com frequência se as documentações ou mensagens compartilhadas não estão corretas e alinhadas com os reais objetivos e necessidades do projeto, portanto a importância de especificar os requisitos corretamente e reforçar o entendimento sobre eles entre os *stakeholders* é muito grande.

As principais causas que resultam em problemas na modelagem de requisitos são: falta de clareza sobre os objetivos a serem atingidos ou qual problema é necessário resolver; gastos com funcionalidades nunca utilizadas ou desnecessárias; complexidade desnecessária da especificação (BITTNER, 2009). Estes fatores vão em encontro aos impactos causados e apresentados no estudo apresentado no primeiro parágrafo (VERNER; SAMPSON; CERPA, 2008).

Os estudos e artigos citados nos parágrafos anteriores reforçam os impactos que ocorrem quando não há a preocupação em modelar os requisitos e critérios de aceitação do *software* de um modo que o entendimento sobre o que precisa ser feito é comum aos *stakeholders*, e também quando não há uma gestão efetiva das comunicações.

* 1. Metodologia

A metodologia utilizada neste trabalho é uma pesquisa de natureza qualitativa, realizada após a aplicação de uma lista de práticas em um projeto real de desenvolvimento de *software*.

Para o desenvolvimento deste trabalho, as seguintes etapas foram realizadas:

1. Definição do escopo do trabalho e motivações para sua realização.
2. Levantamento bibliográfico e estudo de material relacionado à prática de definição de comportamentos esperados utilizando BDD e notação *Gherkin*, verificando como isso auxilia na comunicação entre as partes envolvidasno projeto, bem como estudo do tópico de Gestão das Comunicações do Projeto presente no PMBOK®.
3. Criação de uma lista com práticas de Gestão das Comunicações, considerando documentos de requisitos e plano de testes no formato proposto pelo BDD.
4. Aplicação dos itens desta lista em um projeto real de desenvolvimento de *software*.
5. Avaliação da influência das práticas na finalização do projeto mediante um questionário.

1.5 Estrutura do Trabalho

O Capítulo 1 INTRODUÇÃO apresenta as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 REVISÃO BIBLIOGRÁFICA apresenta os principais conceitos relacionados ao trabalho, que são Metodologias Ágeis, XP, TDD, *Behaviour-Driven Development* e Gestão das Comunicações dentro do proposto pelo PMBOK®.

O Capítulo 3 PROPOSTA E APLICAÇÃO DE PRÁTICAS PARA GESTÃO DAS COMUNICAÇÕES COM BDD apresenta uma lista de práticas, focadas em refinamento de requisitos mediante definição de comportamentos com BDD e gestão das comunicações. Esta lista será aplicada em um projeto real de desenvolvimento para verificar se suas proposições terão influência positiva no sucesso do projeto.

O Capítulo 4 ANÁLISE DE RESULTADOS irá verificar, através de um questionário aplicado aos participantes do projeto, se as práticas propostas contribuíram para uma especificação de requisitos e gestão das comunicações eficientes. O *software* entregue também será parâmetro para avaliar os resultados do trabalho, sendo que o principal critério a ser considerado é a funcionalidade entregue, ou seja, se o *software* faz aquilo que o usuário realmente precisa.

O Capítulo 5 CONSIDERAÇÕES FINAIS descreve a conclusão, contribuições e propostas para trabalhos futuros.

REFERÊNCIAS contém os artigos, livros, monografias e documentos utilizados na pesquisa realizada para elaboração deste trabalho.

2. REVISÃO BIBLIOGRÁFICA

2.1 Considerações Iniciais

Este capítulo fornece o referencial teórico utilizado para realização deste trabalho.

Inicialmente, serão tratados os assuntos: Metodologias Ágeis, *Extreme Programming* e TDD, para que seja possível fornecer um contexto sobre as origens do BDD, que é o tema central do trabalho.

Posteriormente, serão compartilhadas as definições e conceitos do BDD, com foco na definição de requisitos e critérios de aceite com utilização de uma linguagem ubíqua, através da sintaxe *Gherkin*.

Por fim, será tratada a disciplina de Gestão das Comunicações do PMBOK®, para relacionar suas práticas com as técnicas de modelagem de requisitos utilizando BDD, e verificar como essas duas abordagens podem se complementar para facilitar as comunicações e consequentemente o sucesso de um projeto.

2.2 Metodologias Ágeis

Os fundamentos das metodologias ágeis surgiram no Manifesto Ágil (BECK *et al.*, 2001), que compartilha os valores e princípios destas metodologias, sendo uma alternativa às abordagens tradicionais utilizadas na época.

Apesar de representarem uma resposta à burocracia e baixa adaptabilidade à mudanças de modelos como o Cascata, estas metodologias envolvem processos disciplinados e o sucesso na adoção das mesmas estão relacionados aos fatores abaixo (PAULK, 2002):

1. Aplicar as metodologias com fidelidade às filosofias e princípios do desenvolvimento ágil.
2. Implementar a metodologia de forma adequada ao ambiente da organização e características da aplicação a ser desenvolvida.

Através da identificação e compilação de melhores práticas de desenvolvimento que utilizavam, e a fim de colaborar com outros profissionais na tarefa de desenvolver *software* (OLIVEIRA *apud* BECK *et al.,* 2015), os autores apresentaram os seguintes valores (BECK *et al.*, 2001):

1. Indivíduos e interações, mais que processos e ferramentas.
2. *Software* em funcionamento, mais que documentação abrangente.
3. Colaboração com o cliente, mais que negociação de contratos.
4. Responder a mudanças, mais que seguir um plano.

De acordo com os autores, os itens à esquerda devem ser mais considerados e tratados com maior prioridade do que os itens à direita. Isso não implica que os valores à direita não tenham importância, mas direciona as prioridades sobre o que deve ser mais valorizado.

Os autores também apresentaram os princípios que estão por trás do Manifesto Ágil e que são seguidos por eles (BECK *et al.*, 2001), que estão presentes em *frameworks* consolidados como o Scrum (SCHWABER; SUTHERLAND, 2017) e metodologias como o XP (*Extreme Programming*). Este capítulo irá enfatizar o XP devido à relação desta metodologia com as práticas de teste unitário que posteriormente foram evoluídas para o BDD.

2.3 *Extreme Programming*

O *Extreme Programming* (XP), introduzido por Kent Beck, é um conjunto de práticas de desenvolvimento ágil de *software* que enfatiza o trabalho colaborativo (WELLS, 1999) com o objetivo de se obter um produto de maior qualidade e aderente à necessidade dos clientes (OLIVEIRA *apud* BECK, 2015).

Os valores do XP (comunicação, simplicidade, *feedback*, respeito e coragem), incentivam a colaboração entre desenvolvedores, times de negócio e demais envolvidos, promovendo um ambiente de entregas contínuas, transparência e rápida adaptabilidade a mudanças (WELLS,1999), abaixo uma breve descrição destes valores (BECK, 1999):

* Comunicação: uma comunicação precisa, objetiva e oportuna ajuda os envolvidos a entenderem quais são os objetivos do projeto e como alcançá-los.
* Simplicidade: almejar sempre a solução mais simples busca minimizar o impacto causado pelas mudanças e o esforço em realizá-las.
* *Feedback*: obter constantes retornos sobre o funcionamento do *software*, mediante testes unitários, programação em pares e alinhamento com times de negócio.
* Respeito: é necessário ter escuta ativa e levar em consideração as ideias que são mais simples e que levam ao atingimento dos objetivos traçados.
* Coragem: capacidade de aceitar e se adaptar a mudanças.

O ciclo de vida de um projeto utilizando XP passa pelas seguintes fases: Exploração; Planejamento; Iterações para versões; Produção e Manutenção, conforme apresentado na figura 1.

Figura 1 – Fases de um projeto utilizando XP



Fonte: HENRAJANI (2007)

Na fase de exploração, há o levantamento das primeiras histórias de usuário e elaboração da versão inicial do projeto (*design*) de arquitetura. Na fase de planejamento, as estimativas são realizadas e as histórias de usuário que serão implementadas na próxima iteração são definidas. Na iteração, uma pequena versão do produto é codificada e testada (a nível de teste unitário e posteriormente teste de aceitação). As funcionalidades testadas e aprovadas pelo cliente são liberadas para a fase de produção, e as rejeitadas geram novas histórias que serão reconsideradas na fase de planejamento e adicionadas em versões futuras do produto. A fase manutenção implica em responder às mudanças que surgirem ao longo do ciclo de vida (HENRAJANI, 2007).

Dentre as práticas introduzidas, estão aquelas relacionadas à qualidade e propriedade coletiva de código (realizadas na fase de iteração), como programação em pares (promovendo revisão do código e aplicação de melhores práticas de desenvolvimento), desenvolvimento dirigido a testes (TDD, no qual os testes unitários são escritos antes da programação da funcionalidade), refatoração de código, integração contínua e testes de aceitação (WELLS, 2000). A figura 2 apresenta um fluxo com estas práticas de propriedade coletiva de código.

Figura 2 – Ciclo de Propriedade Coletiva de Código no XP



Fonte: WELLS (2000), traduzido pelo autor

Das práticas de propriedade coletiva de código promovidas pelo XP, o trabalho irá explorar o TDD, que é o desenvolvimento dirigido por testes unitários.

2.4 *Test-Driven Development*

O *Test-Driven Development* (TDD; em português, Desenvolvimento Dirigido à Testes) é uma abordagem cíclica de desenvolvimento apresentada por (BECK, 2002) no XP, na qual codificação deve se iniciar pela escrita dos testes, e posteriormente a implementação da funcionalidade que irá fazer aquele teste passar é realizada.

Essa dinâmica visa a obtenção de maior agilidade na entrega das funcionalidades, maior facilidade de manutenção do código e aumento da confiabilidade na solução de modo que os custos com correção de defeitos em produção sejam reduzidos, pois uma base de testes automatizados é gerada e executada a cada nova versão do *software*, antes da publicação para produção.

Apesar destas vantagens, a percepção geral é que o TDD é algo complicado de aprender e aplicar, além da questão contra-intuitiva de que a produtividade, ao invés de melhorar, irá diminuir com o surgimento das novas tarefas de escrita de testes no processo. Além disso, a exigência de mais dedicação e disciplina do que nas práticas de desenvolvimento *test-last* mais tradicionais, reforçam a necessidade de capacitar os profissionais no entendimento e aplicação destas técnicas *test-first* (LATORRE, 2013) para que os benefícios de as utilizar sejam obtidos.

O ciclo do TDD ocorre conforme representado na figura 3.

Figura 3 – Ciclo do Test-Driven Development



Fonte: OLIVEIRA (2015), adaptado pelo Autor

A classificação das tarefas ocorre da seguinte maneira (BECK, 2003):

* Vermelhas: escrita e execução de um teste que vai falhar pois o código não está implementado.
* Verdes: implementação do código de maneira pragmática para o teste que falhou anteriormente passar.
* Refatoração: melhoria do código e evoluções na implementação, com o cuidado de eliminar possíveis duplicidades e más práticas de implementação.

A fim de colaborar com a limpeza dos testes e do código, e para consolidar a aplicação do TDD entre os profissionais de desenvolvimento, Robert C. Martin propôs que a criação dos testes antes da implementação é apenas o primeiro passo para realização desta prática com excelência, e que as três leis do desenvolvimento dirigido à testes devem ser consideradas (MARTIN, 2008):

1. Não se deve escrever o código de produção até criar um teste de unidade de falhas.
2. Não se deve escrever mais de um teste de unidade do que o necessário para falhar, e não compilar é falhar.
3. Não se deve escrever mais códigos de produção do que o necessário para aplicar o teste de falha atual.

Entretanto, conforme já explorado nos parágrafos anteriores, existe a percepção de que a aplicação do TDD é complexa, principalmente em seus estágios iniciais, nos quais os desenvolvedores sentem dificuldades na compreensão do que seria a escrita de um teste que falhe.

Uma das respostas à essas dificuldades foi o BDD (NORTH, 2006), que além das abordagens de testes de aceitação, busca apoiar a compreensão sobre os comportamentos esperados do sistema, de modo que seja possível tirar proveito dessa compreensão não somente nas atividades de testes, mas também de refinamento de requisitos.

2.5 *Behaviour-Driven Development*

O *Behaviour-Driven Development* (BDD; em português Desenvolvimento Dirigido a Comportamentos), é um conjunto de práticas apresentado em um artigo de Dan North publicado em 2006, no qual o autor propõe uma evolução das práticas do TDD, com o objetivo de minimizar as dificuldades de entendimento dos desenvolvedores sobre o quê deve ser testado, de que forma aquilo deve ser testado ou mesmo porquê um teste falhou.

Por focar na especificação e testes dos comportamentos que o sistema deve apresentar, essas práticas aproximam os desenvolvedores dos times de negócios e testes, visto que a linguagem utilizada é comum para as partes (ubíqua) e o entendimento do quê deve ser feito e de como o sistema deve se comportar é equalizado. Isso também serve como apoio sobre quais funcionalidades (ou seja, comportamentos) devem ser priorizadas nas entregas de valor do projeto (NORTH, 2006).

Além das boas práticas para escrita de testes, como nomear a assinatura do método de teste com uma frase com o comportamento esperado, no BDD há a proposta de que requisitos também são comportamentos, com inspiração no conceito de linguagem ubíqua proposta nas abordagens do *Domain-Driven Design* – DDD (EVANS, 2004).

Os princípios do BDD afirmam que os times de negócio e tecnologia devem se referir ao sistema, no que diz respeito às suas terminologias e comportamentos, da mesma forma, e que esse entendimento é permitido justamente pela definição de uma linguagem comum entre as partes (LAZĂR; MONTOGNA; PÂRV, 2010). Esta definição permite que:

1. Os requisitos sejam especificados através de uma perspectiva de negócio.
2. Os analistas de negócio definam exemplos concretos a fim de esclarecer o comportamento do sistema.
3. Os desenvolvedores implementem as funcionalidades através de uma abordagem dirigida à testes (ou seja, escrevam os testes primeiro).

A fim de criar uma sintaxe que colabore com o conceito e aplicação de uma linguagem ubíqua, o autor buscou inspiração em um formato de história de usuário utilizado na época (NORTH, 2006), no qual os critérios de aceite estavam incorporados à escrita da história, conforme:

Como <ator>

Quero <alguma funcionalidade>

Para <atingir um objetivo>

Com estas bases, uma estrutura de especificação de cenários foi proposta (NORTH, 2006) em uma sintaxe que depois veio a ser conhecida como *Gherkin* (OLIVEIRA; MARCZAK, 2017).

Uma especificação de um cenário utilizando *Gherkin* tem a seguinte sintaxe (OLIVÉRIO, 2016):

Dado algum contexto inicial (em inglês, “Given”)

Quando um evento ocorre (em inglês, “When”)

Então, garanta alguns resultados (em inglês, “Then”)

Além das palavras reservadas “Dado”, “Quando” e “Então”, também está presente na sintaxe o comando de sentenças adicionais “E” (em inglês, “*And*”), caso seja necessário incluir mais pré-condições, eventos ou pós-condições (ZEN, 2014).

Para identificação da funcionalidade e cenários, também são utilizadas palavras reservadas.

Um cenário que pode ser utilizado como exemplo para demonstrar esta sintaxe é o de saque de dinheiro em um caixa eletrônico, conforme proposto por Dan North no artigo que apresentou o BDD (NORTH, 2006). Um fator relevante sobre a escrita de cenários é a de que eles devem contemplar os casos de sucesso e de falhas, dentro de uma mesma funcionalidade. O exemplo compartilhado neste tabalho é o de um caso de sucesso, os exemplos na íntegra estão disponíveis no artigo de North.

Funcionalidade: Cliente saca dinheiro

Cenário: A conta está com saldo positivo

Dado que a conta que está com saldo positivo,

E o cartão seja válido,

E o dispensador contenha dinheiro,

Quando o cliente requisita dinheiro

Então, garanta que a conta seja debitada,

E garanta que o dinheiro seja dispensado,

E assegure-se que o cartão seja devolvido.

Além disso, indo ao encontro à definição de que os analistas de negócios devem fornecer exemplos concretos para esclarecimento dos comportamentos do sistema (LAZAR; MONTOGNA; PÂRV, 2010), é possível utilizar tabelas de dados apresentando possíveis valores e demais informações que possam ser relevantes para o entendimento dos comportamentos esperados (WYNNE; HELLESØY, 2017), em qualquer uma das três expressões principais (Dado, Quando e Então).

Para a escrita dos cenários especificando os comportamentos esperados e posterior automação dos testes de aceitação, recomenda-se o uso de ferramentas como o *Cucumber* (WYNNE; HELLESØY, 2017), que tem suporte à sintaxe *Gherkin*.

Da própria documentação oficial do *Cucumber*, foi extraído um exemplo do uso de tabela de dados para explicitar valores e comportamentos esperados (SMARTBEAR SOFTWARE©, 2019):

Dado que os seguintes usuários estejam cadastrados no sistema:

| Nome | Email | Twitter |

| Aslak | aslak@cucumber.io | @aslak\_hellesoy |

| Julien | julien@cucumber.io | @jbpros |

| Matt | matt@cucumber.io | @mattwynne |

As próprias recomendações sobre como utilizar o *Cucumber* e seu propósito maior de validar o entendimento mútuo dos envolvidos sobre como o *software* deve se comportar reforçam que as práticas do BDD, antes de qualquer coisa, se referem à alcançar este entendimento, de modo que a automação dos testes seja consequência (HELLESØY, 2015), conforme citação direta de Aslak Hellesøy, criador do *Cucumber*:

*Cucumber* não é uma ferramenta de testar *software*. É uma ferramenta para testar o entendimento das pessoas em como um *software* (ainda a ser escrito) deve se comportar.

Este entendimento pode ser utilizado para modelar os requisitos funcionais de um *software*, e aliados à alguns conceitos de escrita de histórias de usuário através da técnica INVEST (COHN, 2004), podemos obter uma padronização ainda maior sobre a escrita dos cenários bem como uma avaliação qualitativa dos envolvidos à respeito das especificações.

Inspirado pelo INVEST, um estudo definiu 14 fatores para escrita de cenários em BDD com qualidade, que são: atômico; completo; consistente; conciso; estimável; viável; independente; negociável; priorizável; pequeno; testável; intelegível; não-ambíguo e valioso (OLIVEIRA; MARCZAK, 2017).

Diante dos conceitos apresentados nesta seção, o enfoque será na utilização do BDD como mecanismo de modelagem dos requisitos para equalizar o entendimento mútuo entre os times de desenvolvimento e negócios dos comportamentos desejados do *software*, criando uma base de conhecimento cujo acesso e utilização podem ser amplificados utilizando técnicas de Gestão das Comunicações, que é o próximo tópico abordado neste trabalho.

2.6 Gestão das Comunicações

2.6.1 O processo de comunicação

De forma genérica, o conceito de comunicação surgiu do termo em latim *communicare*, que significa “tornar comum” (MAZUCA *apud* HARPER, 2018), cujo objetivo é transmitir um propósito expresso em alguma mensagem ou informação e torná-lo comum entre as partes envolvidas no processo de comunicação. Para tanto, é necessário no mínimo um emissor e um receptor desta mensagem.

A mensagem é codificada por seu emissor para algum formato textual ou simbólico, e é transmitida através de alguma mídia (meio de comunicação) até o receptor, que realiza o entendimento (decodificação) da mensagem iniciada pelo emissor e gera um feedback deste entendimento ao receptor. Neste processo, também podem surgir os ruídos, que são quaisquer interferências que comprometam o entendimento do receptor da mensagem ou mesmo a produção da mesma por seu emissor (SONDERMANN *et. al.*, 2013). A figura 4 apresenta estes elementos do processo de comunicação em um diagrama de fluxo de dados.

Figura 4 – Elementos presentes no processo de comunicação



Fonte: SONDERMANN *et. al.* (2013)

2.6.2 Fluxos de Gestão das Comunicações

A Gestão das Comunicações é uma das áreas de conhecimento do PMBOK®, que inclui os processos necessários para assegurar que as informações do projeto sejam planejadas, coletadas, criadas, distribuídas, armazenadas, recuperadas, gerenciadas, controladas, monitoradas e finalmente organizadas de maneira oportuna e apropriada.

Esta área de conhecimento engloba 3 processos:

* Planejar o Gerenciamento das Comunicações: desenvolve uma abordagem e um plano adequado para execução das atividades relacionadas à Gestão das Comunicações, com base nas necessidades de informações dos *stakeholders* e do projeto em si. A figura 5 apresenta fluxo do planejamento da Gestão das Comunicações.

Figura 5 – Fluxo do Planejamento da Gestão das Comunicações



Fonte: PMBOK® (2017)

* Gerenciar as Comunicações: assegura a coleta, criação, distribuição, armazenamento, recuperação, gerenciamento e disposição final das informações do projeto. A figura 6 apresenta fluxo do Gerenciamento das Comunicações.

Figura 6 – Fluxo do Gerenciamento das Comunicações



Fonte: PMBOK® (2017)

* Monitorar as Comunicações: garante que as necessidades de informação do projeto sejam atendidas. A figura 7 apresenta fluxo de Monitoração do Gerenciamento das Comunicações.

Figura 7 - Fluxo de Monitoração do Gerenciamento das Comunicações



Fonte: PMBOK® (2017)

Ainda segundo o PMBOK®, a informação pode ser transmitida de forma escrita, falada, de maneira formal (documentos formais da organização) ou informal (mídias sociais ou aplicativos de troca de mensagens) e através de mídias (imagens e vídeos, por exemplo). Para a comunicação de forma escrita, uma das mais típicas, o guia recomenda o uso dos 5 C’s:

* Correta: evitar o uso deficiente da ortografia e gramática, pois isso pode causar distrações e diminuir a credibilidade da mensagem.
* Concisa: redigir a mensagem de forma eficiente, para evitar ambiguidades que possam comprometer o entendimento.
* Clara: garantir um propósito clarro e direcionado às necessidades do receptor.
* Coerente: utilizar um fluxo lógico, coerente e fluído de ideias.
* Controlada: controlar o fluxo de palavras, se utilizar de imagens e resumos, quando aplicável.

Outras habilidades em comunicação podem potencializar o efeito de mensagens escritas considerando os 5 C’s, como: escuta ativa; consciência de diferenças sociais e pessoais entre os envolvidos e identificação, definição e gerenciamento de expectativas das partes interessadas. Também podemos nos valer de práticas consideradas tendências na Gestão das Comunicações para minimizar os ruídos ao longo do projeto, como a inclusão das partes interessadas em revisões e reuniões de projetos.

2.6.3 Gestão das Comunicações em ambientes ágeis ou adaptativos

Em ambientes dinâmicos e fortemente sujeitos a mudanças (o que é uma tendência nos projetos de *software*), há o reforço de que existe uma grande necessidade de se realizar as comunicações com eficiência e rapidez. Com isso, a recomendação do PMBOK® é a dinamização do acesso às informações, pontos de verificação frequentes e, se possível, agrupamento dos membros da equipe / projeto no mesmo local. Também há a necessidade de publicar as documentações do projeto de forma transparente e realizar constantes revisões com as partes interessadas.

2.7 Considerações do Capítulo

O BDD tem como seu principal propósito proporcionar um entendimento mútuo entre os times de desenvolvimento e negócios sobre quais são os comportamentos esperados de um sistema, através da utilização de uma linguagem comum entre as partes (LAZĂR; MONTOGNA; PÂRV, 2010).

Este entendimento é particularmente útil para minimizar problemas causados por tomadas de decisão sem entendimento pleno dos requisitos, um dos ofensores típicos relacionados à finalização mal-sucedida de um projeto (VERNER; SAMPSON; CERPA, 2008).

A busca pela linguagem comum, ou ubíqua, inspirada pelos conceitos de Desenvolvimento Dirigido a Domínios (EVANS, 2004), é representada no BDD através da sintaxe *Gherkin*, tipicamente utilizada em ferramentas como o *Cucumber*, que permite a escrita e automação de cenários nesta sintaxe(WYNNE; HELLESØY, 2017).

Mesmo com a utilização destas ferramentas de automação, é importante ressaltar que a ênfase em modelar cenários através do BDD é validar o entendimento dos envolvidos sobre quais comportamentos o sistema deve ter antes mesmo de sua implementação, a automação dos testes de aceitação é uma consequência deste entendimento, conforme enfatizado pelo próprio criador do *Cucumber* (HELLESØY, 2015).

Dado que é possível e recomendável que as abordagens do BDD sejam propostas como práticas de Engenharia de Requisitos, podemos avaliar se a escrita e entendimento dos cenários está adequada, de uma forma inspirada pelo INVEST (COHN, 2004), mas que consiste em 14 fatores que, se satisfeitos, implicam no entendimento mútuo dos cenários modelados entre os analistas de negócio, desenvolvedores e testadores de *software* (OLIVEIRA; MARCZAK, 2017). Os 5 C’s para uma comunição escrita de qualidade (PMI, 2017) também é um aliado para modelagem destes cenários.

Também é importante reforçar que, por mais que a especificação esteja correta, é preciso assegurar que ela seja conhecida, compreendida e esteja disseminada à todos os envolvidos, ainda mais em um contexo no qual equipes diferentes e distribuídas estão atuando no mesmo projeto (HINDS; BAILEY, 2003), pois isso pode causar uma série de problemas ou mesmo conflitos devido à expectativas divergentes (WILLIAMS; O’REILLY, 1998).

Aliado à isso, um estudo demonstrou que quanto piores as comunicações, maiores os custos de um projeto e sua propensão a falha (PMI, 2013). Estes riscos podem ser diminuídos se utilizarmos as abordagens de Gestão das Comunicações propostas no PMBOK®, que podem ser utilizadas como apoio para disseminação das documentações produzidas, atacando dois dos pontos que levam projetos à serem finalizados com falha: problemas na especificação de requisitos e nas comunicações.

3. PROPOSTA E APLICAÇÃO DE PRÁTICAS PARA GESTÃO DAS COMUNICAÇÕES COM BDD

Este capítulo consiste na elaboração e aplicação de um *Checklist* de práticas de Gestão das Comunicações, em um projeto real de *software*, com atuação de mais de uma equipe, em uma empresa cujo nome será preservado.

As especificações dos requisitos do sistema serão realizadas no formato BDD, e vão passar por uma avaliação qualitativa mediante uma questão do *Checklist* que trata os 5 C’s da comunicação.

O objetivo é validar a influência destes processos e técnicas no apoio às comunicações, entendimento sobre o quê deve ser feito, diminuição de conflitos entre times e consequentemente na finalização do projeto. Estas avaliações, também qualitativas, serão realizadas mediante questionário aplicado aos participantes do projeto após sua finalização.

3.1 Proposta de *Checklist*

A fim de realizar as validações propostas neste trabalho, um *Checklist* de práticas de Gestão das Comunicações e especificação de requisitos com BDD será proposto e aplicado em um projeto de desenvolvimento de *software*.

O projeto consiste na realização de transferências bancárias (tanto entre instituições financeiras diferentes quanto entre contas da própria instituição), no qual serão enfatizados os requisitos referentes à comprovação destas transferências (geração e consultas internas dos comprovantes das transferências efetuadas).

Para atender à esses requisitos, temos a alocação de 2 times: a equipe do sistema de comprovantes e a equipe que irá implementar o *front-end* de consultas em uma estação administrativa corporativa.

Como o *core* desta funcionalidade está no sistema de comprovantes, a equipe deste sistema será classificada como a responsável por prover a funcionalidade, de modo que o time do *front-end* seja classificado como envolvido.

Este Checklist será aplicado pelo Gerente do Projeto e respondido, em conjunto, pelos desenvolvedores, PO’s (ou algum representante de negócio equivalente) e engenheiros de qualidade (QA), dos times responsáveis e envolvidos.

Na ausência de um Gerente de Projeto, o mesmo pode ser aplicado por um facilitador ou Líder de Projeto a ser eleito pelo time.

Além da responsabilidade de aplicar o *Checklist*, o Gerente precisa facilitar a realização e assegurar que todas as cerimônias e processos que envolvam a resposta, manutenções e disseminações do *Checklist* em si e das documentações que derivaram dele sejam cumpridos. Na condução deste trabalho, o autor exerceu a função de Gerente.

A tabela 1 representa o *Checklist* que será aplicado para validação do entedimento dos requisitos e Gestão das Comunicações.

Tabela 1 – *Checklist* de Especificação dos Requisitos

|  |  |
| --- | --- |
| 1. Checklist de Especificação de Requisitos | |
| 1.1 Identificador do Requisito / História | [id único] |
| 1.2 Descrição do Requisito / História | [breve descrição, a especificação em si estará no formato BDD] |
| 2. Identificação Responsáveis | |
| 2.1 PO ou Representante de Negócio | [nome e e-mail] |
| 2.2 Engenheiro de Qualidade (QA) | [nome e e-mail] |
| 2.3 Desenvolvedor | [nome e e-mail] |
| 3. Identificação Envolvidos | |
| 3.1 Domínio do sistema mantido pelo time envolvido | [descrição do domínio do sistema, exemplo: Transferências Bancárias] |
| 3.2 PO ou Representante de Negócio | [nome e e-mail] |
| 3.3 Engenheiro de Qualidade (QA) | [nome e e-mail] |
| 3.4 Desenvolvedor | [nome e e-mail] |
| 4. Refinamento dos requisitos entre os responsáveis | |
| 4.1 Houve reunião de refinamento e especificação de requisitos entre os responsáveis? | [marcar com "x" para sim, deixar em branco para não] |
| 4.2 A especificação de requisitos está em formato BDD? | [marcar com "x" para sim, deixar em branco para não] |
| 4.3 As boas práticas de comunicação (5 C's) foram utilizadas na especificação? | [marcar com "x" para sim, deixar em branco para não] |
| 4.4 Os cenários escritos estão de acordo com os 14 fatores? | [marcar com "x" para sim, deixar em branco para não] |
| 5. Refinamento dos requisitos com os envolvidos | |
| 5.1 Houve reunião de refinamento e especificação de requisitos com os envolvidos? | [marcar com "x" para sim, deixar em branco para não] |
| 5.2 O entendimento dos requisitos foi normalizado junto aos envolvidos? | [marcar com "x" para sim, deixar em branco para não] |
| 5.3 A especificação de requisitos está armazenada em um repositório centralizado, com fácil acesso pelos responsáveis e envolvidos? | [marcar com "x" para sim, deixar em branco para não] |
| 6. Plano de comunicação | |
| 6.1 Há a necessidade de realizar reuniões de checkpoint do projeto? | [marcar com "x" para sim, deixar em branco para não] |
| 6.2 Existe um plano de comunicação / alinhamento a ser aplicado para o caso de mudanças? | [marcar com "x" para sim, deixar em branco para não] |
| 6.3 Tanto os responsáveis quanto os envolvidos possuem conhecimento deste plano? | [marcar com "x" para sim, deixar em branco para não] |

Fonte: Autor.

3.1.1 Considerações sobre a aplicação do *Checklist*

O *Checklist* apresentado na seção anterior se refere à uma validação sobre a qualidade da especificação de requisitos em formato BDD, se fazendo valer de técnicas de Gestão das Comunicações. Diante desse cenário, ele é aplicado no momento de se realizar o refinamento dos requisitos.

O termo “refinamento” pode ser interpretado como a fase de análise e desenho em abordagens mais tradicionais, e equivale ao *Refinement Meeting* (antigo *Backlog Grooming*) no Scrum, podendo ser incorporado à outras metodologias ágeis (BALDINI, 2019).

Esta reunião é realizada para reforçar o entendimento dos requisitos priorizados e documentá-los de forma que os mesmos estejam prontos para serem desenvolvidos (SEDANO; RAPLH; PÉRARE, 2019). No modelo proposto, PO, Desenvolvedor e QA realizam esta reunião em conjunto, com a atuação do Gerente de Projeto como facilitador, e o entregável da mesma é a especificação das funcionalidades priorizadas em formato BDD. Este alinhamento e especificação permite com que as 4 primeiras seções do *Checklist* sejam respondidas.

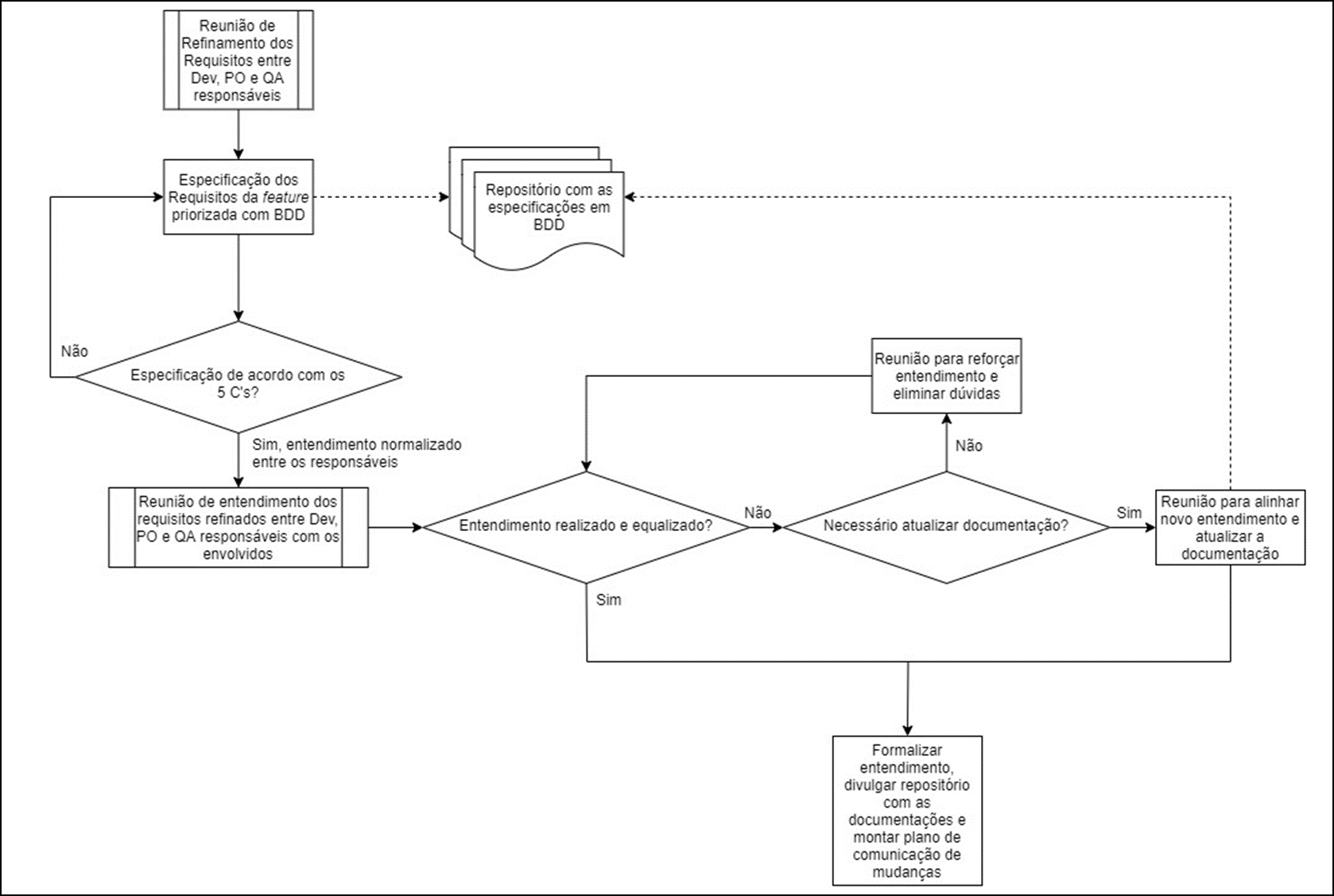
Após esse primeiro alinhamento, as documentações produzidas são apresentadas ao time envolvido em uma reunião (neste caso, o sistema de comprovantes irá fornecer uma API de consulta às comprovações de transferências para o sistema envolvido), na qual dúvidas são tiradas e eventuais ajustes nas especificações são realizados.

Caso o entendimento não tenha sido equalizado nesta conversa, as dúvidas ou pendências são anotadas, e um novo alinhamento é marcado com o retorno destes pontos.

Uma vez que esse entendimento é concluído entre as partes, as questões de plano de comunicações e resposta à mudanças são respondidas e os desenvolvimentos podem ser iniciados.

A figura 8 ilustra os procedimentos compartilhados nesta seção em um fluxograma.

Figura 8 – Fluxo de processos para preenchimento do *Checklist*



Fonte: Autor

3.y Considerações do Capítulo

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

4. ANÁLISE DE RESULTADOS

[Este capítulo apresenta uma análise crítica do capítulo anterior. Pode ser complementado com capítulo adicional através de um Estudo de Caso ou Experimento de uma aplicação prática. Colocar títulos adequados]

4.x Título da Seção

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

4.y Considerações do Capítulo

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

5. CONSIDERAÇÕES FINAIS

[Este capítulo descreve as conclusões e as contribuições do trabalho e sugestões de trabalhos futuros que poderão ser desenvolvidos a partir deste trabalho]

5.1 Conclusões

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

5.2 Contribuições do Trabalho

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

5.3 Trabalhos Futuros

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

REFERÊNCIAS

[Para mais detalhes consultar a referência abaixo]

Divisão de Biblioteca – EPUSP. Diretrizes para Apresentação de Dissertações e Teses da USP. 4ª. Edição. São Paulo. 2020. 75p.

<https://doi.org/10.11606/9788573140651>

PAULK, Mark C., Agile Methodologies and Process Discipline. Carnegie Melon Software Engineering Institute, 2002.

GLOSSÁRIO

[As descrições a seguir foram extraídas de bla bla bla bla bla.]

*[Termo 1]* – bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

*[Termo 2]* – bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

Apêndice A – [Título do Apêndice]

[Texto ou documento elaborado pelo autor]

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.

ANEXO 1 – [Título do Anexo]

[Texto ou documento não elaborado pelo autor]

Bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla bla.