UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA
ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Angelita Anderle

Introdução de BDD (Behavior Driven Development) como Melhoria de Processo no Desenvolvimento Ágil de Software

São Leopoldo
2015

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS

UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA

ESPECIALIZAÇÃO EM QUALIDADE DE SOFTWARE

Angelita Anderle

Introdução de BDD (Behavior Driven Development) como Melhoria de Processo no Desenvolvimento Ágil de Software

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Especialista em Qualidade de Software, pelo curso de Pós-Graduação Lato Sensu em Qualidade de Software da Universidade do Vale do Rio dos Sinos – UNISINOS.
Orientador: Prof. Me. Odisnei Galarraga

São Leopoldo
2015

Introdução de BDD (Behavior Driven Development) como melhoria de Processo no Desenvolvimento Ágil de software

Angelita Anderle[1]

[1]Unidade Acadêmica de Educação Continuada - Universidade do Vale do Rio dos Sinos (Unisinos) - São Leopoldo – RS - Brasil

aanderle@gmail.com

Abstract. *This paper presents the results and conclusions of the introduction of the technique of BDD (Behavior Driven Development) and user stories, as process improvement in a project that is using scrum. The research-action was used as a method of research, data collection via questionnaires answered by project members and by observation, in addition to the evaluation of content analysis. Through data analysis, the results showed how positive was improving in agile software development using BDD.*

Resumo*. Este artigo apresenta os resultados e conclusões da introdução da técnica de BDD (Behavior Driven Development) - Desenvolvimento Dirigido por Comportamento - e de histórias de usuários (User Stories), como melhoria de processo em um projeto que utiliza scrum. Utiliza-se pesquisa-ação como método de pesquisa, coleta de dados via questionários com os membros do projeto e pela observação, além da avaliação da análise de conteúdo. Realizada a análise dos dados, os resultados mostraram o quanto foi positiva a melhoria no desenvolvimento ágil de software utilizando o BDD.*

1. Introdução

De acordo com Helm e Wild (2014), por mais que as tecnologias evoluam, o processo de desenvolvimento ainda é complexo. Cohn (2004, p3) afirma: "Requisito de software é um problema de comunicação".

Frequentemente, observa-se que profissionais envolvidos em desenvolvimento de software em algum momento já fizeram menção à representação mostrada na figura 1, extraída de https://flavioaf.wordpress.com que ilustra a construção de um balanço em uma árvore, solicitado pelo cliente, e mesmo este tendo passado por todas as etapas do projeto, ao final detectou-se uma falha de comunicação/entendimento.

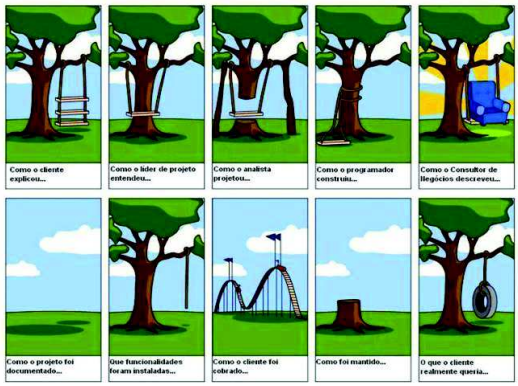

Figura 1. Dificuldade do entendimento claro do requisito.

Oppermann (2014) ressalta o resultado obtido pela PriceWaterhouseCoopers, onde após um estudo com cerca de 200 empresas de vários países, identificou-se que apenas 2,5% delas obtiveram sucesso nos últimos anos em relação aos seus projetos. E ainda levanta a questão destas empresas focarem demais nas práticas e processos ao invés de focarem também nas pessoas envolvidas e na clareza de comunicação.

Pressman (2011) também destaca a falta de qualidade de software, fazendo com que as empresas tivessem uma grande preocupação em relação a uma falha no software que poderia provocar problemas de infraestruturas e gerar um aumento no custo em dezenas de bilhões. A questão mais importante, é que muitas empresas americanas, conforme CIO Magazine, estavam gastando ou desperdiçando bilhões em softwares que não estavam fazendo o que supostamente deveriam fazer.

Koch e Oliveira (2010), após terem realizado um trabalho no qual foi conduzida uma série de entrevistas com o intuito de identificar os "Riscos na Utilização de Métodos ágeis na gestão de Projetos de Software", destacaram:

> "Todos entrevistados possuem experiência prática com gerenciamento de projetos ágeis utilizando Scrum. Sobre o time, que são as pessoas responsáveis pela construção do software, os maiores riscos levantados pelos entrevistados são relacionados à comunicação. MA (Métodos Ágeis) pregam a comunicação como um dos seus valores primários, quando não executado de forma adequada pode ser um dos principais fatores para o fracasso". (p12)

Neste contexto, fica visível o valor das empresas possuírem uma boa técnica em relação aos requisitos (todos entenderem o que realmente o cliente solicitou) que permita resolver estas questões de modo a não perderem clientes por não conseguirem entregar o sistema, ou ainda, entregar o sistema e este não estar de acordo com o que deveria supostamente fazer.

Este artigo sugere e descreve alterações no processo de requisitos, expressos em termos de histórias de usuário da área de análise e desenvolvimento, com o objetivo de responder as seguintes questões: Com o uso da técnica de BDD é possível ter um melhor entendimento dos requisitos? Quais os ganhos obtidos com essa técnica? Como poderá ajudar todos do projeto a ter um entendimento melhor dos requisitos e do que deverá ser desenvolvido?

De acordo com Smart (2015), temos vários problemas que muitas vezes só são descobertos na fase de qualificação e que poderiam ser evitados com o uso do BDD, pois se detectaria o problema ainda na fase de requisitos. Com o BDD, é estabelecida uma linguagem comum onde todos os membros do projeto têm um mesmo entendimento do que foi solicitado.

Segundo BECK, Kent et al, (2001) o Manifesto Ágil enfatiza a interação e comunicação entre os indivíduos. Scrum é uma de várias metodologias de desenvolvimento ágil que depende de uma boa comunicação entre o time e uma boa resposta a mudanças para a entrega de um produto de software que atenda às necessidades do cliente (Pressman, 2011).

O objetivo geral deste trabalho é propor e implementar uma melhoria no processo de desenvolvimento, usando BDD, para propiciar um melhor entendimento dos requisitos em um projeto Scrum da empresa "XYZ". Para atingir o objetivo geral, foram definidos alguns objetivos específicos: a) Mapear o processo de desenvolvimento de um projeto *Scrum* que não utilize BDD; b) Identificar os pontos de melhoria onde o BDD poderia

contribuir; c) Propor as modificações necessárias no processo com a introdução de BDD; d) Implementar as melhorias propostas e analisar seus efeitos nos resultados do processo, nesse contexto organizacional; e) Identificar, com os participantes de projetos que utilizaram o processo que incluiu BDD, os eventuais benefícios que o uso dessa técnica lhes proporcionou.

O presente trabalho será aplicado na empresa "XYZ" que é uma multinacional instalada no Rio Grande do Sul, cuja sede está localizada nos Estados Unidos. A empresa atua na área de desenvolvimento de software para diferentes produtos e segmentos e possui mais de 3 milhões de usuários que utilizam o software por ela desenvolvido.

Além da introdução, este artigo está organizado em mais 5 seções. A seção 2 apresenta o referencial teórico para o desenvolvimento do trabalho. A seção 3 apresenta a metodologia aplicada, delineamento da pesquisa, contexto da melhoria, técnica para a coleta de dados e análise dos dados. A seção 4 apresenta a melhoria do processo, contexto do estudo, situação atual, alteração do processo e a análise dos dados. Por último, a seção 5 traz as conclusões finais do estudo.

2. Referencial Teórico

Para contextualizar o assunto abordado, são apresentados nesse capítulo alguns conceitos fundamentais de desenvolvimento e qualidade de software, bem como os conceitos relacionados ao BDD.

2.1 Desenvolvimento Ágil

Segundo Sommerville (2011), nos dias atuais as empresas trabalham em ambientes globais e com mudanças rápidas. Desta forma, é necessário ter uma resposta breve para as novas oportunidades e novos mercados. Sommerville (2011, p38) ressalta que: "o desenvolvimento e entrega rápidos são, portanto, o requisito mais crítico para o desenvolvimento de sistemas de software".

Sendo assim, Audy (2013) levanta a questão do "Manifesto Ágil", focado em entender como os projetos lidam com imprecisão e imprevisibilidade. O Manifesto Ágil, enunciado em 2001, por Kent Beck e mais dezesseis renomados consultores, enumera pontos e boas práticas para contribuir com a melhoria nos processos de desenvolvimento.

Esse manifesto vem ao encontro da valorização do trabalho do indivíduo para com o desenvolvimento (BECK, Kent et al):

- **Indivíduos e interação entre eles** mais que processos e ferramentas
- **Software em funcionamento** mais que documentação abrangente
- **Colaboração com o cliente** mais que negociação de contratos
- **Responder a mudanças** mais que seguir um plano

Sendo que não é negado o valor dos itens à direita, mas é atribuído mais valor aos itens à esquerda.

Podemos também destacar os 12 princípios por trás do Manifesto Ágil:

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência em menor escala de tempo.

4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe e desenvolvimento é através de conversa face a face.

7. Software funcionando é a medida primária de progresso.

8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.

10. Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.

11. As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis.

12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Pressman argumenta que:

> "Se os membros da equipe de software devem orientar as características do processo que é aplicado para construir software, deve existir um certo número de traços-chaves entre as pessoas de uma equipe ágil e a equipe em si: " (2011, p86)

Para que a metodologia funcione, os fatores humanos são muito importantes, conforme Pressman (2011) relaciona:

Competência: Dentro do contexto ágil, todos os membros da equipe se posicionam de acordo com as suas habilidades, e devem e podem ensinar o conhecimento para outros membros da equipe.

Foco Comum: Por mais que todos os membros da equipe façam diferentes tarefas, todos devem ter em mente um único objetivo comum. Por exemplo, entregar no prazo acordado que o time se comprometeu.

Colaboração: Fazer uso de informações comunicadas tendo como foco o entendimento de todos os membros da equipe para a realização das tarefas.

Habilidade na tomada de decisão: A equipe deve ter autonomia para controlar e gerenciar suas decisões, tanto referentes a assuntos técnicos como no nível de projeto.

Habilidade de solução de problemas confusos: Em certos momentos, a equipe terá que lidar com ambiguidade e com o fato de que será continuamente atingida por mudanças. Talvez o problema solucionado hoje não fará mais sentido amanhã. Entretanto, a equipe

deverá saber lidar com essas situações e tirar lições disso, pois no futuro poderão ser benéficas.

Confiança mútua e respeito: Uma equipe consistente demonstra a confiança e o respeito necessários para torná-la "tão fortemente unida que o todo fica maior do que a soma das partes". Pressman (2011 p. 86) apud. DeMarco (1998).

Auto-Organização: No contexto ágil, a auto-organização resulta em três fatores: (1) a equipe ágil se organiza para o trabalho a ser desenvolvido, (2) a equipe ordena o processo para melhor se adaptar ao ambiente, (3) a equipe organiza seu cronograma para entregar o seu produto dentro do prazo. O maior benefício da auto-organização talvez venha a ser a colaboração entre os membros da equipe, mantendo-a sempre unida e motivada.

2.2 SCRUM

De acordo com Pham (2012), o termo Scrum surgiu em um artigo publicado por Hirotaka Takeuchi e Ikujiro Nonaka na Harvard Business Review de 1986, denominado "The new new product development game" (O novo jogo do desenvolvimento de novos produtos). Os autores compararam esse estudo das equipes de projeto à formação Scrum do rugby. O rugby é um esporte coletivo, onde um tipo de jogada chamada scrum é utilizada para o reinício do jogo. Com base nesta ideia, Jeff Sutherland, no início dos anos 90, juntamente com sua equipe de desenvolvimento, criou uma metodologia para o desenvolvimento ágil batizando-a com o nome de Scrum (Pressman, 2011).

Pham (2012) reforça que a equipe scrum deveria ser multifuncional, e possuir de cinco a nove membros, com somente três papéis: a) o *Scrum Master*- um facilitador que assegura que o time respeite e siga os valores e as práticas do método; b) o *Product Owner* - representante do cliente que ajudará a identificar os requisitos e tomará as decisões referentes ao negócio; c) a equipe de desenvolvimento (time), com todas as aptidões necessárias para desenvolver o produto solicitado.

Pressman (2011) apresenta que o Scrum tem seus princípios consistentes com o manifesto ágil, onde seu alicerce contempla as atividades estruturais: requisitos, análise, projeto, evolução e entrega do produto/funcionalidade, onde essas atividades ocorrem dentro de um padrão chamado sprint. A Sprint é um ciclo que pode variar entre 2 a 4 semanas (dependendo da complexidade do produto) e contempla uma quantidade de requisitos/histórias para serem implementadas. Essas user stories (histórias de usuários), como são chamadas, serão definidas pelo cliente durante a fase de levantamento de requisitos e passarão a fazer parte da lista do Product Backlog (Backlog do produto) como pode ser representada na figura 2 abaixo, extraída de http://blog.fasagri.com.br/?p=125.

Conforme Griebler (2011), todas as requisições e atividades estarão sob o controle do PO (*Produc Owner*) e formam o chamado *Product Backlog* (*backlog* do produto), que será uma lista priorizada de requisitos, onde vários aspectos podem compor essa lista, desde aspectos do negócio e tecnologia, questões de arquitetura e correções de *bugs*. Então, esta lista dará origem ao *Sprint Backlog* que será desenvolvido.

Figura 2. Ciclo do Scrum

2.3 Qualidade de Software

Este tópico não tem a pretensão de apresentar todos os pontos sobre qualidade de software, mas somente os aspectos relevantes para o presente trabalho.

Segundo Pressman (2011), a qualidade de software teve um maior destaque no momento que o software passou a ficar mais integrado com as atividades do nosso cotidiano. Na década de 90, teve-se um reconhecimento por parte das organizações de que bilhões de dólares estavam sendo desperdiçados em software, e que este não estava sendo apresentado e se comportando conforme o esperado.

Pressman ainda questiona: tendo um software de má qualidade, qual seria o prejuízo?

> "...especialistas dizem que bastam três ou quatro defeitos a cada 1.000 linhas de código para fazer com que um programa execute de forma inadequada. Acrescente a isso que a maioria dos programadores insere cerca de um erro a cada 10 linhas de código escrito, multiplicados por milhões de linhas de código em vários produtos comerciais. Assim, deduz-se que o custo dos fornecedores de software será de pelo menos a metade dos seus orçamentos para a realização dos testes e correção dos erros. Percebeu o tamanho do problema? " (2011, p 357)

A questão é que se deve fazer correto da primeira vez, ou será necessário refazer tudo. Se a equipe de software ressaltar a qualidade em todas as suas atividades de engenharia de software, isso fará que com que a quantidade de retrabalho seja reduzida. Com isso, teremos um custo menor e o produto será disponibilizado em um prazo menor.

"Qualidade não é uma fase do ciclo de desenvolvimento de software... é parte de todas as fases." (Bartié 2002, p25)

Uma maneira de garantir que o trabalho foi realizado de forma correta é observar a qualidade por meio da verificação dos resultados das atividades de controle de qualidade, efetuando a verificação de erros antes da entrega e de defeitos que acabaram passando despercebidos e direcionados para produção. (Pressman, 2011)

Matté (2011) comenta que a qualidade de software não se restringe apenas ao produto entregue estar em conformidade com o projeto, da mesma forma que as operações metodológicas aplicadas no processo de desenvolvimento acompanhem padrões.

Glass argumenta que a qualidade é importante, mas se o usuário não estiver satisfeito, nada mais importa. DeMarco reforça esse ponto de vista ao afirmar: "A qualidade de um produto é função do quanto ele transforma o mundo para melhor". Pressman (2011 p. 359) apud. Glass e DeMarco (1998).

Pressman (2011, p 361) apresenta os fatores de qualidade de McCall, Richards e Walters (1977) que criaram uma categorização dos princípios que influenciam a qualidade de software também representada pela figura 3:

Correção: o quanto um programa satisfaz e cumpre os objetivos solicitados pelo cliente.

Confiabilidade: o quanto se pode esperar que um programa realize a função pretendida com a precisão exigida.

Eficiência: a quantidade de recursos computacionais e de código exigida para que um programa execute sua função.

Integridade: o quanto o acesso ao software ou dados por pessoas não autorizadas pode ser controlado.

Usabilidade: o quanto de esforço é necessário para aprender, operar, preparar a entrada de dados e interpretar as saídas produzidas pelo sistema.

Facilidade de manutenção: o quanto de esforço é necessário para localizar e eliminar problemas um programa.

Flexibilidade: o quanto de esforço é necessário para modificar um programa em operação.

Testabilidade: o quanto de esforço é necessário para testar um programa de modo a garantir que ele desempenhe a função determinada.

Portabilidade: o quanto de esforço é necessário para transferir o programa de um ambiente de hardware e/ou software para outro.

Reusabilidade: o quanto um programa (ou partes dele) pode (m) ser reutilizado (as) em outros programas.

Interoperabilidade: o quanto de esforço é necessário para incorporar um sistema a outro.

Figura 3. Fatores de qualidade de software de McCal

2.4 Teste de Software em Métodos Ágeis

Crispin e Gregory (2009), no que se refere ao modelo tradicional, afirmam que enquanto parte dos profissionais de análise de negócio, arquitetura e gestão trabalhavam no desenvolvimento dos requisitos, o testador gerava vários documentos para verificar se os requisitos estavam de acordo com o que foi solicitado. O Modelo V de Testes foi um guia por muito tempo neste ponto, trazendo quatro níveis: requisitos, análise, arquitetura e codificação. Neste ponto, os testadores também criavam os casos de teste, quando o time já estava na fase "codificação". Neste caso, o analista de teste é reativo ao invés de proativo. Muitas vezes o que acontece na prática é que o time de testes acaba trabalhando separadamente da equipe de desenvolvimento e só irão se unir no momento da fase de validação do modelo tradicional.

Teste ágil é um conjunto de práticas, seguindo o Manifesto Ágil, que incorpora todas as técnicas de teste comumente utilizadas por profissionais de teste, tendo um grande foco em automação. A principal função é de criticar o produto, para garantir que o que está sendo especificado e desenvolvido realmente atende as necessidades do cliente e irá entregar valor ao negócio.

Crispin e Gregory (2009) destacam que no desenvolvimento ágil o analista de teste passa por uma transformação que fará com que ele trabalhe proativamente e também uma maior interação com os desenvolvedores e analistas de negócios.

O Quadrante de Teste Ágil (figura 4) foi gerado por Brian Marick, que introduziu uma série de termos para diferentes categorias de teste, elevando a participação e posterior qualidade do profissional que os executa.

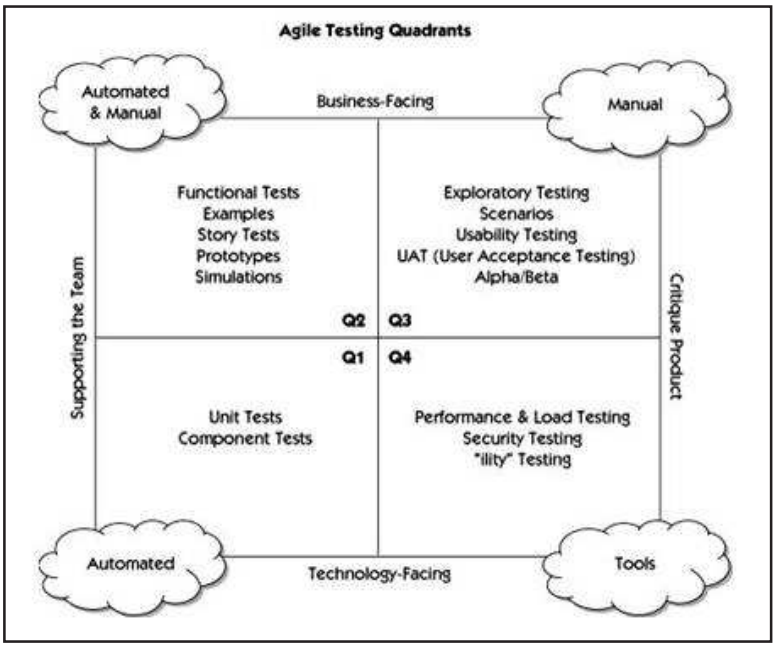

Figura 4. Quadrante de testes ágeis

Conforme Crispin e Gregory (2009) cada quadrante representa:

Quadrante1:

O Quadrante 1 representa o desenvolvimento orientado a testes, a principal prática de desenvolvimento ágil: TDD – Test Driven Development. Neste quadrante temos duas práticas: teste de unidade, que valida um pequeno subconjunto da aplicação como objetos e/ou métodos, e testes de componente, que valida partes maiores da aplicação como um grupo de classes que fornecem algum serviço. Geralmente são desenvolvidos com ferramentas xUnit (nome genérico para qualquer estrutura de testes automáticos unitários), medem a qualidade interna do produto e podem ser automatizados. Com a técnica de TDD o programador pode desenvolver uma funcionalidade sem se preocupar em posteriormente modificar parte da aplicação, auxiliando nas decisões de arquitetura e design. Não são direcionados ao cliente, pois o cliente não irá compreender os aspectos internos do desenvolvimento e não devem ser negociáveis com o cliente, pois a qualidade do código deve constantemente existir e não ser negligenciada. Os testes desenvolvidos são geralmente executados dentro de um comportamento de Integração Contínua para prover em um curto tempo um feedback da qualidade do código

Quadrante 2:

Os testes no quadrante 2 também apoiam a equipe de desenvolvimento, mas de uma maneira mais alto-nível, focando mais em testes que o cliente entenda (negócio), onde este define a qualidade externa e as características que os clientes precisam. Os testes do quadrante 1 também dirigem o desenvolvimento, mas em um nível mais alto. Com desenvolvimento ágil, estes testes são derivados de exemplos fornecidos pelo cliente e descrevem os detalhes de cada história. Testes focados no negócio igualmente podem ser automatizados e a técnica de BDD – Behavior Driven Development - é utilizada na escrita e execução automatizada destes cenários. Além disso, podemos ter uma pessoa de *UX (User eXperience)* para que, através de *mockups* e *wireframes*, o cliente possa validar a interface gráfica antes que a equipe comece a desenvolver esta funcionalidade. Portanto, o foco desses testes não é encontrar o maior número de defeitos e sim ajudar clientes e desenvolvedores a se entenderem melhor.

Quadrante 3:

Criando diversos mecanismos para assegurar a necessidade do cliente por meio de critérios ou exemplos, isto não nos assegura que realmente entendemos o desejo do cliente, e que o teste unitário e o teste via BDD podem não apresentar o real valor. Este quadrante apresenta a questão de criticar o produto e executá-lo como um usuário real, usando nosso conhecimento e instinto na utilização da aplicação. O cliente pode realizar este tipo de tarefa, usualmente chamada de UAT – *User Acceptance Testing*, dando um feedback mais exato, aceitando a funcionalidade, analisando possíveis novas funcionalidades. Esta ação pode ser também um dos critérios de DoD – *Definition of Done* de uma funcionalidade. No quadrante 3, além do UAT, os testes exploratórios são muito importantes. Utilizando esta técnica, qualquer membro da equipe pode aprender sobre a aplicação e executar mais testes, usando o feedback do último teste para a execução dos próximos e também é apto de extrair novos critérios, sempre observando o comportamento da aplicação.

Quadrante 4:

Neste quadrante, os testes são os mais técnicos e criticam o produto em termos de desempenho, carga e segurança. Nos dias atuais, negligenciar aspectos como desempenho pode tirar a vantagem competitiva de um cliente. Geralmente já conhecemos aspectos relacionados a desempenho e segurança quando refinamos algumas histórias de usuários. As técnicas aplicadas a desempenho, carga e segurança vão desde os níveis mais baixos como a utilização de ferramentas que simulam diversos usuários simultaneamente.

Todas as técnicas de testes apresentadas podem ser aplicadas, pois o quadrante é um guia e não um padrão.

2.5 BDD

BDD (Behavior Driven Development – Desenvolvimento Guiado por Comportamento) auxilia as equipes a concentrar seus esforços na identificação e compreensão das histórias de usuários com a visão focada na comunicação e entendimento do requisito no projeto. Smart (2015).

Conforme Auvray (2009), o BDD (*Behavior Driven Development*) é uma resposta ao *Test Driven Development* (TDD) e foi apresentado por Dan North em 2003. O BDD incentiva a colaboração de todos que participam em um projeto de desenvolvimento de software, sendo eles: desenvolvedores, QA (*Quality Assurance*), PO (*Product Owner*), e analistas não técnicos, analista de negócio e arquiteto de software.

BDD evoluiu de práticas ágeis estabelecidas e foi criado para fazê-las mais acessíveis e efetivas para equipes novas em desenvolvimento ágil de software. Por algum tempo, BDD evoluiu até compreender um quadro mais amplo de análise ágil e automação de testes de aceitação. North (2003)

Soares I (2011), diz que o BDD serve para criar testes e integrar regras de negócios com a linguagem de programação, focando no comportamento do software. Em relação aos testes, estes influenciam o desenvolvimento, ou seja, primeiro se escreve o teste e depois o código.

Soares I (2011) ainda destaca que:

> "O foco em BDD é a linguagem e as interações usadas no processo de desenvolvimento de software. Desenvolvedores que se beneficiam destas técnicas escrevem os testes em sua língua nativa em combinação com a linguagem ubíqua (linguagem estruturada). Isso permite que eles foquem em por que o código deve ser criado, ao invés de detalhes técnicos, e ainda possibilita uma comunicação eficiente entre as equipes de desenvolvimento e testes. "

North (2003) sugeriu que a linguagem utilizada no BDD possa ser extraída nas especificações fornecidas pelo cliente durante o levantamento dos requisitos. Isso facilitará a comunicação e entendimento do time como um todo. Ainda ressalta que o *template* não poderia ser engessado e nem artificial ao ponto de criar restrições aos analistas, mas estruturado suficiente de modo a quebrar a história em seus fragmentos constituintes e automatizá-las. Os critérios de aceitação foram descritos em termos de cenários que assumiram a seguinte forma:

Given - Dado algum contexto inicial.
When - Quando algum evento ocorrer.
Then - Então verificar os resultados.

Para demostrar, será usado o exemplo de um caixa eletrônico. Uma história pode parecer com isto:

+Titulo: Cliente faz um saque+

Como um cliente,
Eu quero retirar dinheiro no caixa eletrônico,
Para que eu não precise ficar esperando na fila do banco.

Existem vários cenários a considerar: a conta pode ser em crédito ou não, a conta pode estar negativa, ou já estar no limite de cheque especial. Vários cenários são possíveis, como se o saque vai exceder o limite, ou se o caixa eletrônico não tem dinheiro suficiente.

Usando o *template* dado acima, podemos ter esse exemplo de cenário:

Critérios de aceitação (apresentado como cenário)

+ **Cenário 1**: A conta tem dinheiro +

Dado que há dinheiro na conta
E o cartão é válido
E o caixa eletrônico tem dinheiro
Quando o cliente solicitar dinheiro
Então garantir que a conta será debitada
E garantir que o dinheiro seja entregue
E garantir que o cartão seja devolvido

Pode-se usar o "E" para repetir o último comando múltiplas vezes.

De acordo com JR (2012), o BDD funciona porque "se apoia no uso de um vocabulário pequeno e bem específico. Dessa forma, minimiza 'ruídos' na comunicação de forma que todos os interessados – tanto de TI, quanto do negócio – estejam alinhados."

Smart (2015) reforça que o BDD ajuda as equipes a concentrar seus esforços na identificação, compreensão e na construção de aplicações que agreguem valor para o negócio, e certifica-se que estas aplicações são bem projetas e bem implementadas.

2.5.1 Benefícios do BDD

De acordo com Smart (2015) os benefícios destacados são:

Redução do desperdício: BDD foca o esforço de desenvolvimento na descoberta e entrega das funcionalidades (*features)* que irão fornecer valor ao negócio. Quando a equipe cria uma aplicação que não está alinhada com os objetivos de negócio implícitos do projeto, esse esforço foi desperdiçado para o negócio. Da mesma forma, quando uma equipe escreve uma aplicação que o negócio precisa, mas de uma forma que não é útil para o negócio, a equipe terá que refazer o trabalho para ser ajustado ao que era esperado, resultando em mais desperdício de tempo e esforço.

BDD ajuda a evitar este tipo de desperdício de esforços, ajudando as equipes a se concentrarem em funcionalidades que estão alinhados com os objetivos de negócio. E

também permitindo obter feedback mais útil dos usuários, ajudando as equipes a fazerem mudanças mais cedo se forem necessárias.

Comunicação: BDD encoraja analistas de negócios, desenvolvedores de software e testadores a colaborar mais de perto, permitindo-lhes expressar os requisitos de uma maneira mais testável, numa forma que tanto a equipe de desenvolvimento como de negócios e ainda as partes interessadas, podem facilmente compreender.

Redução de custo: A consequência direta de reduzir tempo é a redução custos. Ao concentrar-se em desenvolver a aplicação com valor de negócio (construindo o software de forma correta), e não desperdiçar esforço em recursos de pouco valor, pode-se reduzir o custo de entregar um produto viável para seus usuários. E melhorando a qualidade do código da aplicação, você reduzirá o número de erros e, portanto, o custo de corrigir estes erros, bem como o custo associado aos atrasos causados por esses erros.

Mudança mais fácil e segura: BDD torna consideravelmente mais fácil de mudar e estender suas aplicações. A documentação é gerada a partir das especificações executáveis, usando termos com os quais as partes interessadas (*stakeholders*) estão familiarizadas. Isso torna muito mais fácil para as partes interessadas compreender o que a aplicação de fato faz. As especificações executáveis de baixo nível também atuam como documentação técnica para desenvolvedores, tornando mais fácil para eles entenderem a base de código existente e fazerem suas próprias alterações. E as práticas de BDD produzem um conjunto amplo de testes de aceitação automatizados e unitários, o que reduz o risco de regressões causado por qualquer nova alteração na aplicação.

***Releases* mais rápidas:** Esses testes automatizados abrangentes também aceleraram o ciclo da *release* notadamente. Testadores não são mais obrigados a realizar longas sessões de testes manuais antes de cada nova release. Em vez disso, eles podem usar os testes de aceitação automatizados como ponto de partida, e gastar seu tempo de forma mais produtiva e eficiente em testes exploratórios e outros testes manuais não triviais.

Soares I (2011) também destaca algumas vantagens em usar o BDD:

Comunicação entre equipes: em algumas empresas de desenvolvimento de software nem sempre os desenvolvedores e testadores trabalham de uma forma unida, mas o BDD proporciona uma aproximação entre os membros da equipe em função dos testadores serem aptos a escreverem os cenários de teste para a equipe;

Compartilhamento de conhecimento: tendo testadores e desenvolvedores trabalhando mais próximos, a tendência é que com o passar do tempo ocorra uma troca de conhecimento, fazendo com que se tenha uma equipe multifuncional;

Documentação dinâmica: algumas equipes ágeis admitem que não documentam a aplicação em função de se tornar custosa. Usando frameworks de BDD, esta documentação será criada dinamicamente. Alguns geram relatórios em formato HTML, o que irá auxiliar uma consulta futura;

Visão do todo: Fergus O'Connell, em sua obra *How to Run Successful High-Tech Project-Based Organizations* (Artech House, 1999), apresenta uma relação dos principais motivos que levam projetos de software ao fracasso. O primeiro deles é: "os objetivos do projeto não são bem definidos e compartilhados entre todos os envolvidos". Por este

motivo, BDD sugere que os analistas/testadores escrevam os cenários antes mesmo dos testes serem implementados, e desta forma os desenvolvedores terão uma visão geral do objetivo do projeto antes de codificá-lo.

North (2003) afirma que:

"Comportamento" é uma palavra mais útil do que "teste": é preciso ter o comportamento bem definido e então o teste passa a ser uma validação destes cenários.

BDD traz uma "linguagem única" para análise;

O comportamento de uma história é simplesmente o seu critério de aceitação;

Os testes são focados no que realmente tem valor para o usuário;

BDD tenta ajudar os desenvolvedores no foco sobre o real valor a ser agregado ao cliente;

Rocha (2013) em seu artigo: *Scrum e BDD: o casamento perfeito*, ressalta:

"Evitam-se erros de compreensão e interpretação das histórias de usuário;

O que antes eram requisitos funcionais e requisitos não funcionais são transformados em comportamentos funcionais do software e critérios de aceitação;

Fornece uma forma de comunicação (vocabulários) comum a todos envolvidos, facilitando a compreensão por parte de todos;

A equipe fica ciente dos comportamentos que serão aceitos pelos usuários com antecedência;

A equipe entende como os comportamentos são relacionados, evitando confusão;

As reuniões de planejamento, revisão e diárias tornam-se mais eficazes. "

2.6 Requisitos e Histórias de Usuários

Helm (2014, p3) afirma que "qualidade de software começa na especificação".

Segundo Helm e Wild (2014), sendo o desenvolvimento de software ainda muito complexo apesar de toda tecnologia, temos as seguintes fases envolvidas:

- ✓ Análise de negócios;
- ✓ Análise de requisitos;
- ✓ Projeto de banco de dados;
- ✓ Desenvolvimento;
- ✓ Testes;
- ✓ Implantação.

De acordo com a realidade da organização o processo terá uma complexidade diferente. Porém, duas fases são primordiais no desenvolvimento de software: especificação e desenvolvimento.

Para Helm e Wildt (2014, p7) “O desenvolvimento de software começa na fase de análise, e principalmente na especificação dos requisitos”.

Para a equipe, não é suficiente ter desenvolvedores extremamente competentes se a especificação/requisitos não tiver todas as informações necessárias, ou seja, for incompleta ou ainda superficial. No entanto,

> “ é totalmente possível especificar software de uma forma muito mais efetiva, simples e até divertida do que o mercado normalmente tem feito ao logo dos anos. Essa forma de especificação de requisitos mais eficiente se chama Histórias de Usuário (*user stories*), que é uma pratica ágil de desenvolvimento de software” (2014, p7)

O desenvolvedor encontra formas diferentes de desenvolver uma mesma funcionalidade. Mas, para que isso seja uma decisão acertada, quanto mais informações ele tiver, melhor, pois não adianta apenas saber “o que fazer”, mas sim “para quem” será a nova funcionalidade criada. Na verdade, também é muito importante que ele saiba o “porquê” da funcionalidade ser desenvolvida e essas questões irão auxiliá-lo a tomar decisões mais assertivas com o que é esperado pelo cliente, e ainda terá um ganho expressivo na qualidade.

2.7 Melhoria de Processo de Software

De acordo com Oliveira (2011), algumas motivações para a busca da qualidade do processo de software são o aumento da qualidade do produto, a diminuição do retrabalho, maior produtividade, a redução do tempo para atender o mercado, uma maior competitividade e maior previsão nas estimativas. Um bom processo não assegura que os produtos produzidos são de boa qualidade, mas é um indicativo de que a organização é capaz de produzir bons produtos.

“O principal inimigo de uma empresa desenvolvedora de software é a baixa qualidade”. Até então, nenhuma pessoa desenvolveu uma proposta válida para melhorar a qualidade que não tivesse relação com a melhoria de processos, a qual passa a ser o ponto principal. As empresas podem ter ferramentas (de software) e pessoas qualificadas, mas são os processos que de fato tornam possível o crescimento de produtividade. (Boria et al.,2013, p1).

Sommerville (2011) destaca a busca constante das organizações por softwares mais baratos e melhores, e que estes sejam entregues em prazos menores. Assim sendo, muitas empresas de software tenderam para a melhoria de processo como a condição de melhorar a qualidade de software, reduzindo custos ou acelerando seus processos de desenvolvimento. A melhoria de processos envolve o entendimento dos processos existentes e sua alteração para aumentar a qualidade de produtos e/ou diminuir custos e o tempo de desenvolvimento.

Duas formas distintas são abordadas para a melhoria e a mudança de processos:

- A abordagem de **maturidade de processo** concentra-se em melhorar o gerenciamento de processos e projetos e em inserir boas práticas de engenharia de software. O nível de maturidade de processos retrata o grau em que as boas práticas técnicas e de gerenciamento foram aceitas nos processos de desenvolvimento de software.

- Já a **abordagem ágil** concentra-se no desenvolvimento iterativo e na diminuição de *overheads* gerais no processo de software. A entrega rápida da aplicação e a habilidade de lidar com mudanças de requisitos dos clientes são as principais particularidades desta

abordagem. E ainda focam no código que está sendo gerado e reduzem, deliberadamente, a formalidade e a documentação.

Em todas as organizações, sendo elas grandes ou não, os processos de software podem ser verificados. Esses processos são de tipos diferentes, pois isso vai derivar do grau de formalidade dos processos, tipos dos produtos criados, tamanho da organização, etc. Não existe a proposta de um processo de software "ideal" ou "padrão" que possa de alguma maneira ser aplicado a todas as organizações ou para todos os produtos de software de um modo específico.

"Cada empresa precisa desenvolver seu próprio processo dependendo de seu tamanho, de seu *background* e das habilidades de seu pessoal, do tipo de software que esteja sendo desenvolvido, do cliente e dos requisitos de mercado, bem como da cultura da empresa". Sommerville (2011, p. 495)

3. Metodologia

Considerando que o objetivo deste trabalho foi de propor e implementar uma melhoria no processo de desenvolvimento, fez-se necessário definir a metodologia utilizada.

3.1 Delineamento da Pesquisa

Para o presente trabalho foi utilizada a abordagem de pesquisa qualitativa. Godoy (1995) fala que o caráter qualitativo é dado quando se têm questões ou focos de interesse amplos, que vão se definindo à medida que o estudo se desenvolve; portanto, justifica-se a abordagem qualitativa. Ainda segundo Godoy (1995), quando a preocupação for a compreensão da teia de ralações sociais e culturais que se estabelecem no interior das organizações, o trabalho qualitativo pode oferecer evidências interessantes e relevantes.

Para Lewin (1946), a Pesquisa-Ação, surgiu da necessidade de superar a lacuna entre teoria e prática, que tem como objetivo promover uma intervenção e melhorias em uma situação atual que está sendo estudada como parte do processo de pesquisa, através da ação do pesquisador. É a produção de conhecimento guiada pela prática. A figura 5 representa o modelo de ciclo de vida da pesquisa-ação segundo Lewin.

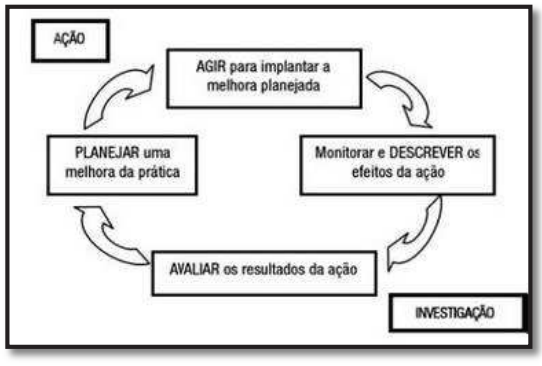

Figura 5. Ciclo Pesquisa Ação (Lewin)

Esta pesquisa-ação foi realizada em uma empresa "XYZ", onde se escolheu projetos que trabalhassem com métodos ágeis (*scrum*) com o objetivo de propor e implantar uma melhoria no processo de desenvolvimento com o uso de BDD. Como resultado, os dados de pesquisa serão comentados em relação ao êxito ou não dos objetivos propostos.

A figura abaixo apresenta uma visão simplificada da descrição dos passos metodológicos utilizados para a realização do trabalho.

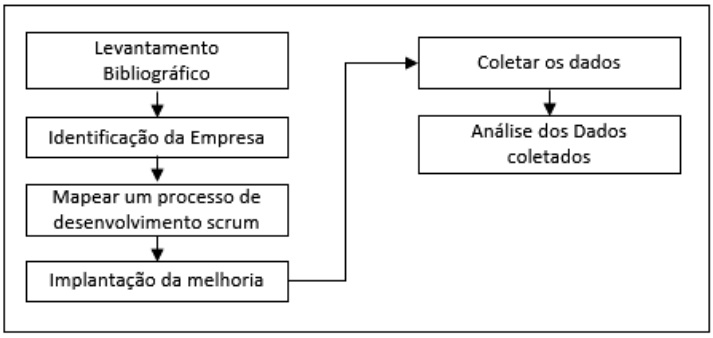

Figura 6. Passos metodológicos

Levantamento Bibliográfico: Para a elaboração do estudo proposto, fez-se um levantamento bibliográfico que serviu para estabelecer uma base conceitual para os termos utilizados neste trabalho, bem como para os fatores que levaram a criação do questionário;

Identificação da Empresa: Conforme já foi abordado no item 1.4, uma empresa que utiliza métodos ágeis em seu desenvolvimento e Scrum foi selecionada para o presente estudo;

Mapear um processo de desenvolvimento Scrum: Através do mapeamento de um desenvolvimento Scrum, verificar os pontos de melhoria;

Implantação da melhoria: Como melhoria no processo, será introduzido o BDD na lacuna identificada a ser preenchida;

Coleta de dados: Para o presente estudo, foi elaborado um questionário (10 questões). No item 3.3 (Técnica para coleta de dados) mais detalhes serão apresentados.

Análise dos Dados coletados: Foi realizada a análise dos dados que consistiu em examinar os resultados (capítulo 5).

3.2 Contexto da melhoria

O contexto da melhoria abrangido no estudo faz parte da área de desenvolvimento de software da empresa "XYZ". Este estudo é focado em projetos Scrum, que foram criados para desenvolver e/ou aprimorar uma funcionalidade já existente no sistema.

Este foi selecionado de uma forma conveniente, dado que a equipe é responsável por uma aplicação. Dois projetos da empresa foram escolhidos para o estudo, pois ambos trabalham com Scrum, totalizando 18 pessoas. Na empresa, ainda existe outro projeto que trabalha com *Kanban*, mas este não possui as características desejáveis para ser abordado no estudo pois o seu objetivo é basicamente correções de defeitos vindos de produção. Em relação ao outro projeto que usa scrum, a resposta foi negativa, pois alguns membros do time não querem usar BDD no momento.

O método de amostragem por conveniência não estatístico é muito usado. Não existe um rigor estatístico, o pesquisador seleciona os sujeitos que têm alcance, admitindo que estes sejam capazes de representar um universo. (Moresi, 2003)

3.3 Técnica para a Coleta de Dados

Para desenvolver o estudo, foi utilizada a técnica de observação e ainda elaborado um questionário formado por um conjunto de questões que são respondidas por escrito pelo pesquisado (GIL, 2002).

As questões foram criadas com base no que os autores identificaram como sendo os benefícios do BDD (Item 2.5.1) dentro de um projeto, com o intuito de obter respostas e analisar se estas responderão as questões ao qual o objetivo do trabalho se propõe.

Segundo Gil (1994), é importante determinar o número adequado de perguntas levando em consideração o possível interesse dos respondentes pelo tema. O mesmo autor comenta que "alguns autores estabelecem como regra geral que o número de perguntas de um questionário não deve ultrapassar a trinta" (1994, p. 129).

Passo 1 - Fez um levantamento de quais projetos scrum poderiam ser implementadas as melhorias no processo.

Passo 2 – Foi realizado um primeiro contato pessoalmente com os integrantes do time para sensibilizar os colaboradores sobre a importância deles neste processo de coleta de dados.

Passo 3 – Com o consentimento do colaborador, de posse do questionário gerado em formato de formulário, estes foram enviados via SurveyMonkey.com (Apêndice A). O colaborador, assim, poderia responder eletronicamente ou, se achasse necessário, poderia imprimir e responder de forma manual. Na maioria das questões, se fez necessário um questionamento usando o "Por quê?" tanto para quando a resposta seria "Sim" ou "Não", sendo que este não era obrigatório, mas sim uma resposta aberta.

Passo 4 - Após o envio do questionário aos colaboradores foi estabelecido um prazo de retorno (4 dias), sem a exigência de identificação de nome por respondente no questionário.

Inicialmente, foram considerados 4 projetos para a pesquisa. Entretanto, para a realização do estudo foram abordados 2 desses projetos que usam scrum. Projeto A e Projeto B. Foi levantada a questão de implantar a melhoria no processo para os outros 2 projetos, Projeto C (scrum) e Projeto D (kanban), mas no caso do Projeto C a resposta foi negativa e o Projeto D, não possui as características desejáveis no momento.

Sendo assim, o estudo considerou o Projeto A e o Projeto B, totalizando 18 pessoas, entre elas desenvolvedores, PO (*product owner*), Analistas de testes, arquitetos de software, *Scrum Master*.

O questionário foi respondido individualmente por pessoas inseridas nos projetos A e B que utilizam *Scrum*, ao qual foram submetidas as melhorias usando BDD.

Os participantes possuem um papel fundamental, pois fornecem percepções e interpretações sobre o assunto, assim como fontes para a busca de evidências corroborativas (YIN, 2001).

Questão:	**Referência utilizada:**
1 - Há quanto tempo você trabalha com desenvolvimento de software Ágil?	Elaborada pela autora: com o intuito de saber quanto tempo o colaborador possui de experiência com desenvolvimento ágil.
2 - Qual a sua função dentro da equipe? Desenvolvedor/ QA/PO/ Arquiteto de Software /Scrum Master/Outros	Elaborada pela autora: com o intuito de saber qual a função do colaborador dentro do projeto.
3 - Você vê valor no uso do BDD no seu projeto? Sim / Não. Por quê?	Smart (2015) - o BDD ajuda as equipes a concentrar seus esforços na identificação, compreensão e na construção de aplicações que agreguem valor para o negócio, e certifica-se que estas aplicações são bem projetas e bem implementadas.
4 - No seu projeto a US (*user story*) é escrita em formato de BDD? Se sim, tendo as US escritas no formato BDD, você acredita que isso trouxe um ganho de qualidade no desenvolvimento do software?	Rocha (2013) - Evita-se erros de compreensão e interpretação das histórias de usuário; North (2003) - BDD traz uma "linguagem única" para analise;
5 - Você acredita que com o uso do BDD todos da equipe tiveram um melhor entendimento do que era requisitado? Sim / Não. Por quê?	Rocha (2013) - O que antes eram requisitos funcionais e requisitos não funcionais são transformados em comportamentos funcionais do software e critérios de aceitação; North (2003) - Isso facilitará a comunicação e entendimento do time como um todo.
6 - Você acredita que com o uso do BDD melhorou a comunicação entre os membros da equipe? Sim / Não. Por quê?	Smart (2015) - Comunicação (encoraja analistas de negócios, desenvolvedores de software e testadores a colaborar mais de perto) Soares I (2011) - Comunicação entre equipes.

Questão:	Referência utilizada:
7 - Você acredita que com o uso do BDD o compartilhamento do conhecimento entre os membros da equipe (por ex: QA, DEV, PO, Scrum Master) melhorou? Sim / Não. Por quê?	Soares I (2011) - Compartilhamento de conhecimento. Rocha (2013) - As reuniões de planejamento, revisão e diárias tornam-se mais eficazes.
8 - Quais as vantagens e desvantagens que você vê com o uso do BDD?	Smart (2015) - Releases mais rápidas; North (2003) - Os testes são focados no que realmente tem valor para o usuário;
9 - Relacionado ao esforço do desenvolvimento, você acredita que houve menos retrabalho com o uso do BDD? Sim / Não. Por quê?	Smart (2015) - Redução do desperdício; Redução de custo.
10 - De uma maneira geral, após a introdução e a aplicação do BDD no projeto, você acredita que foi possível transformar os requisitos em um produto que agrega valor ao cliente?	North (2003) - BDD tenta ajudar os desenvolvedores no foco sobre o real valor a ser agregado ao cliente;

Quadro 1: Questionário sobre o uso do BDD em um projeto ágil

3.4 Técnicas de Análise dos Dados

Após o questionário ser enviado e tendo as questões respondidas, foi realizada uma análise destas respostas, que é apresentada no capítulo 5, onde é verificada a eficácia ou não da introdução do uso de BDD no processo de desenvolvimento.

Todas as respostas, a partir da questão 3, encontram-se no Apêndice B. No que se refere às duas primeiras questões, não foi necessária a análise de conteúdo visto que essas não são diretamente questões abertas, pois uma aborda a experiência do colaborador com o desenvolvimento ágil e a outra é relacionada à função exercida no projeto.

As questões são abertas, obtendo respostas dissertativas pelos respondentes. A técnica comumente adotada na análise de respostas qualitativas é a análise de conteúdo.

De acordo com Flick (2009), na síntese da análise de conteúdo, o material é parafraseado, o que indica que termos e paráfrases menos pertinentes que possuam significados iguais são omitidos. E ainda ressalta que esse tipo de análise é clássico para analisar materiais textuais sendo estes gerados através de entrevistas, mídia e outros.

Segundo Moraes (1999), de certa forma, a análise de conteúdo é uma interpretação pessoal por parte do pesquisador vinculado ao entendimento que tem dos dados. Uma leitura neutra não é realizável. Toda leitura atribui-se a uma interpretação.

4 . Melhoria do Processo

Neste capítulo será apresentada a melhoria realizada no processo Scrum utilizado nos projetos pesquisados.

4.1 Contexto do Estudo

O estudo realizado teve como objetivo a identificação de pontos de melhoria em um processo de desenvolvimento de software pela introdução de BDD. Em um segundo momento, foi realizada uma avaliação para identificar se estas alterações trouxeram benefícios para o projeto em termos de comunicação e correta compreensão dos requisitos. A partir dos resultados, espera-se observar se houve ganho de qualidade no desenvolvimento e consequentemente no produto final.

4.2 Situação Atual

Segundo, Sommerville (2011), nesta etapa o processo atual é examinado e os gargalos e pontos fracos são identificados. O processo atual da metodologia de desenvolvimento do time que se quer realizar a introdução de BDD deve ser observado, permitindo assim, a identificação dos pontos de melhoria. O Projeto A e B fazem uso do scrum para o desenvolvimento.

O processo começa com a interação do cliente/*Stakeholders* (parte interessada) com o Dono do Produto (*PO – Product Owner* – quem representa o cliente na empresa que desenvolverá o produto).

Após as partes interessadas definirem as funcionalidades ou alterações desejadas, o PO inclui uma nova entrada na ferramenta usada para auxiliar o gerenciamento ágil dos requisitos pelo *PO*. Essa entrada é chamada de história de usuário. A *US* é criada de forma descritiva podendo conter de 3 a 5 linhas ou mais e não considera nenhum tipo de padrão. No momento em que é criada, a *US* vai para o *Product Backlog* para futura priorização a ser realizada pelo *PO*.

Na reunião de planejamento da *Sprint,* o time se reúne para avaliar as histórias de usuário priorizadas pelo *PO* e adiciona uma pontuação para cada história. Identifica-se se é suportada pelo time para ser desenvolvida na *sprint*, sendo que esta deve ser concluída em 3 semanas no caso dos times selecionados, pois o tamanho da *Sprint* pode variar dependendo do projeto ou de como foi acordado (ex: pode-se ter *Sprints* de 2, 3 ou 4 semanas).

Ao longo da s*print* são realizadas reuniões diárias de curta duração (15 minutos é o recomendado) onde 3 perguntas básicas devem ser respondidas por cada membro do time:

1. O que você fez ontem?
2. O que você fará hoje?
3. Existe algum impedimento que te impeça de atingir teu objetivo?

Durante a *sprint,* podem surgir dúvidas oriundas do desenvolvedor sobre a história de usuário que está em desenvolvimento. A dúvida é levada ao *PO* para que seja esclarecida. Caso seja identificado que é uma história de usuário mais complexa do que foi descrita, esta pode ser removida da Sprint ou dividida. E havendo ausência de

comunicação para melhor entendimento da US e negociação, caso ela seja mais complexa do que inicialmente estimada, pode levar à falha da *sprint*.

Sob o ponto de vista de qualificação, é neste momento também que os casos de testes são criados e também os *scripts* para automação dos mesmos usando a ferramenra *ALM* (Application Lifecycle Management - QTP). A criação dos *scripts* é acordada previamente durante a reunião de planejamento, pois, na realidade do time observado há casos em que não é possível concluir a automação de todas as US na mesma *sprint*.

O time, em conjunto com o PO, formaliza a Definição de Pronto (DoD – *Definition of Done*) para a *US*, que nada mais é do que os passos mínimos para a conclusão de um item potencialmente empregável. Após a conclusão da US, também é realizada uma demonstração para o *PO*, apresentando a funcionalidade requerida.

Após cada sprint é realizada uma reunião de retrospectiva para avaliar os pontos positivos e negativos que ocorreram e os itens de ação que serão tomados. Tendo todas as Sprints concluídas, no final o cliente recebe o que foi desenvolvido e testado. A figura 7 representa o processo scrum obtido de http://www.leanti.com.br/artigos/24/o-que-e-scrum-a-metodologia-sob-o-ponto-de-vista-lean.aspx.

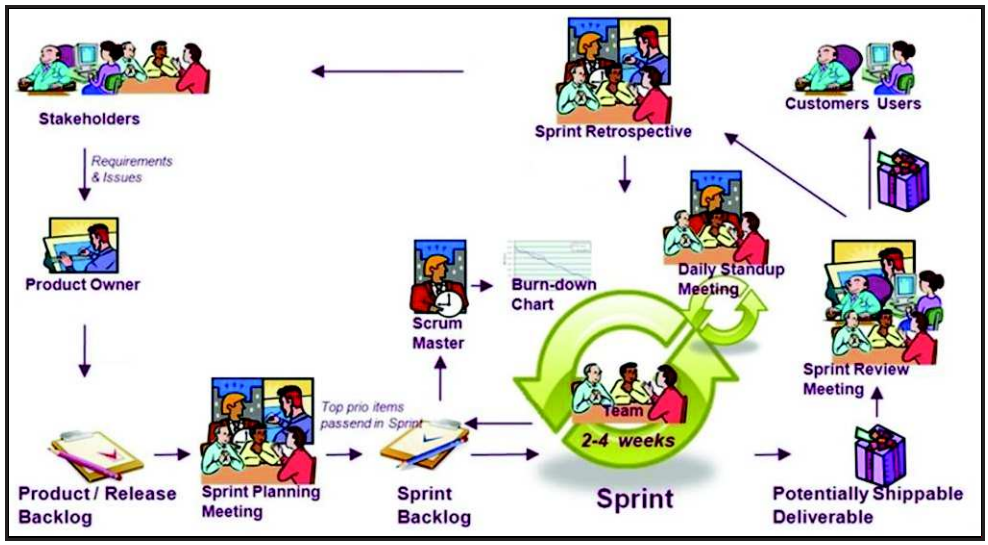

Figura 7. Envolvidos no processo *scrum*

Em frequentes conversas com os participantes dos projetos que usam Scrum, e também por meio de observações relacionadas a eventuais problemas que aconteciam nesses projetos, foram identificados pontos do processo onde poderia ser obtida uma melhoria com a adoção da técnica de BDD. Isto motivou a realização deste trabalho, pois o ambiente mostrou-se propício a uma proposta de introdução dessa mudança.

Os pontos para melhoria identificados foram:

- As histórias de usuários (*US*) são escritas em formato "convencional" (por definição *who, what*, e *why*) ou seja, em alguns casos a descrição é apresentada em três ou quatro linhas, deixando margem para interpretações diferentes por parte dos membros do time. Em relação aos critérios de aceitação, não são descritos (ou pobremente descritos), o que acaba também gerando diferentes interpretações. Isto faz com que haja muitas dúvidas na hora do desenvolvimento, algumas podendo aparecer somente no momento do teste;

- Geração de retrabalho para o desenvolvimento por não se ter cenários e critérios de aceitação claros;

- Mesmo que a metodologia (*agile/scrum*) estimule a discussão dos critérios de aceitação durante o andamento da *Sprint,* observa-se que não é uma prática comum.

- Nem todos os membros do time têm um entendimento do cenário e de como esse deve se comportar, ou seja, não fica claro o que o cliente espera receber no final do desenvolvimento;

- Existe uma perda relacionada ao tempo de criação dos casos de testes manuais ou automatizados, pois estes são criados de acordo com o entendimento da história apresentada e da forma como os cenários são descritos. Não é possível aproveitar essa descrição para adicionar diretamente em uma ferramenta que auxilie na automação de testes, ficando nítido o trabalho duplicado neste ponto.

- Não existe a troca de conhecimento entre os membros do time para discutir sobre os cenários e critérios de aceitação.

4.3 Alteração do Processo

Sommerville (2011 p 497), diz que "as mudanças de processo são propostas para resolver alguns dos pontos fracos identificados. Elas são introduzidas e o ciclo recomeça para a coleta de dados sobre a eficácia das mudanças".

Com a introdução do BDD (*Behavior Driven Development*) nos dois projetos *scrum* A e B, foram efetuadas alterações na forma de escrita da história de usuário. A *US* é apresentada no formato BDD com seus cenários e critérios de aceitação definidos na ferramenta de gerenciamento de requisitos (Rally), com o propósito de melhorar o entendimento do que é requisitado por todos do time.

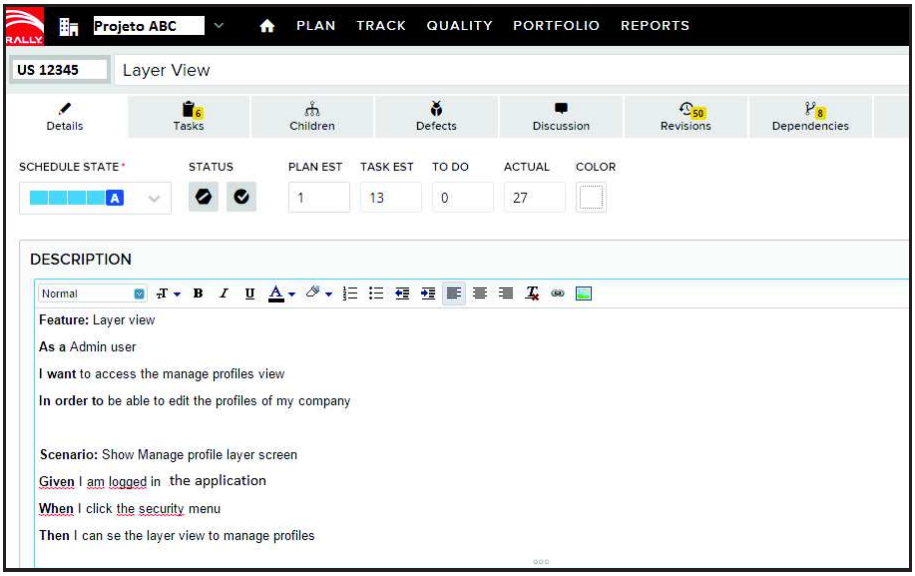

Figura 8. Imagem de uma US na ferramenta Rally

Outra alteração muito importante é que além do *PO* descrever a *US* (história de usuário) no formato BDD, apresentando os cenários e critérios de aceitação, também são realizadas reuniões com o time todo para entender os requisitos e neste momento, se o time encontra algum outro cenário, este é adicionado. Essas reuniões são chamadas de

grooming. Por ter-se uma linguagem comum, isso deve facilitar para o desenvolvedor na construção do software que incide sobre os recursos que realmente importam para o negócio.

Com os critérios de aceitação de uma US descritos em formato BDD, isso pode ser utilizado como entrada para a criação de testes automatizados. Estes cenários são transcritos para o Selenium com o *plugin* do Cucumber, na linguagem Java, auxiliando na execução do teste de regressão. Na verdade, o BDD permite a utilização de outras ferramentas para a automação que podem facilitar e tornar o processo mais dinâmico. O testador também usa os resultados destes testes como ponto de partida para testes manuais e exploratórios.

Segundo Smart (2015), a figura 9 abaixo ilustra um time que usa BDD. No caso do Scrum, temos o papel do PO, do desenvolvedor, testador e ainda arquiteto de software que colaboram para entender e definir os requisitos juntos.

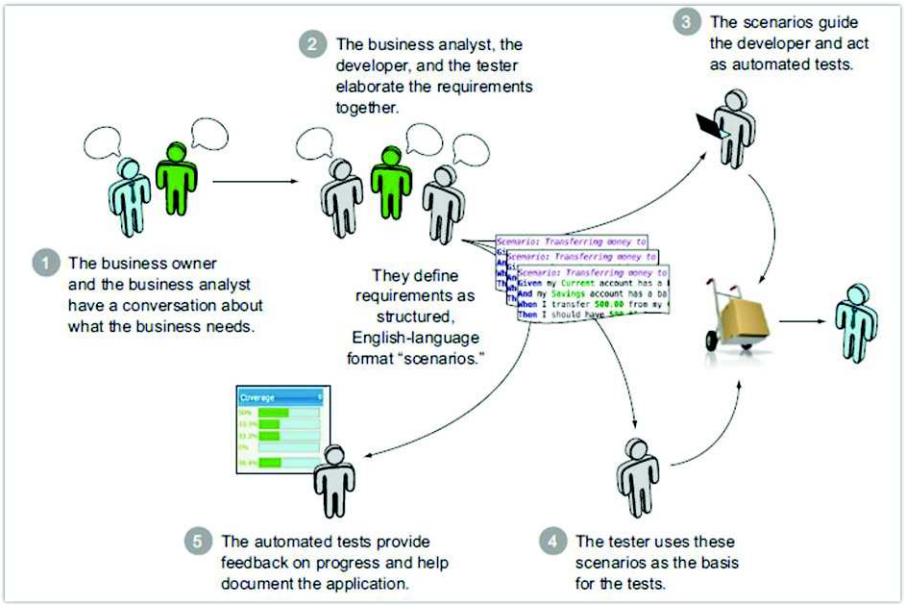

Figura 9. BDD introduzido no processo

Para identificar se com a introdução do BDD no projeto os colaboradores identificaram/perceberam benefícios usando essa técnica, foi aplicado um questionário com as perguntas relevantes a levantar esses resultados. A análise das respostas é apresentada no próximo capítulo.

4.4 Análise das respostas ao questionário de pesquisa

A análise dos dados desta pesquisa foi realizada com o auxílio da ferramenta Microsoft Office Excel. Este passo consistiu em examinar e obter indicativos dos resultados coletados, dadas as proposições iniciais do estudo.

Abaixo são apresentados os resultados obtidos, buscando respostas para as questões de pesquisa deste trabalho. Obteve-se 100% dos questionários respondidos pelos 2 (dois) projetos, totalizando 18 questionários respondidos.

Projeto A possui: 2 POs (está ocorrendo a transição de um deles), 2 desenvolvedores, 1 *scrum master*, 1 arquiteto de software e 2 analistas de testes.

Projeto B possui: 1 PO, 5 desenvolvedores, 1 *scrum master*, 1 arquiteto de software e 2 analistas de testes.

Por questões didáticas, foram invertidas a ordem das questões 1 e 2 para uma melhor apresentação e entendimento das siglas relacionadas às funções.

2 - Qual a sua função dentro da equipe?

- ✓ Dentre os 18 colaboradores, identificou-se que as funções abaixo encontram-se distribuídas nos 2 projetos de desenvolvimento ágil.
  - (7) são Desenvolvedores (**Dev**);
  - (4) são Analistas de Teste (**AT**);
  - (3) são Product Owner (**PO**);
  - (2) são Arquitetos de software (**Arq**) e;
  - (2) são *Scrum Master* (**SM**);

1 - Há quanto tempo você trabalha com desenvolvimento de software Ágil?

- ✓ Identificou-se conforme respostas abaixo, que a média de anos que os colaboradores trabalham com desenvolvimento ágil é de 2,5 anos, o que indica que alguns ainda são iniciantes na metodologia ágil.

SM1	3 anos		AT3	8 meses
SM2	6 anos		AT4	3 anos
Arq1	2 anos		Dev1	8 meses
Arq2	4 anos		*Dev2*	1 ano
PO1	6 anos		Dev3	3 anos
PO2	3 anos		Dev4	1 ano
PO3	4 anos		Dev5	10 meses
AT1	6 meses		Dev6	2 anos
AT2	2 anos		Dev7	2 anos

Quadro 2 - Funções X Anos de trabalho com desenvolvimento ágil

3 - Você vê valor no uso do BDD no seu projeto? Sim / Não. Por quê?

- ✓ Todos participantes afirmaram ver valor no BDD, pois veem ganho na descrição dos cenários de negócio desde o início do desenvolvimento, facilitando o desenvolvimento e teste e evitando retrabalho. Ainda, alguns corroboram com esta visão, citando que é possível colaborar melhorando os requisitos e que proporcionou uma aproximação maior entre os membros do time.

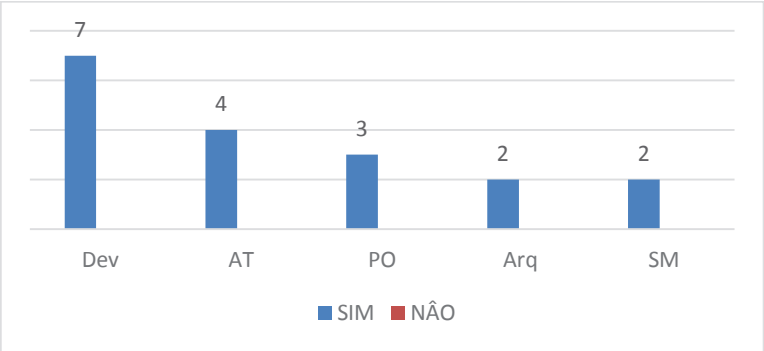

Figura 10 - Participantes do projeto referente a questão 3.

4 - No seu projeto a US (*user story*) é escrita em formato de BDD? Se sim, tendo as US escritas no formato BDD, você acredita que isso trouxe um ganho de qualidade no desenvolvimento do software?

- ✓ Dos 18 participantes, 14 afirmaram que sim, mas em alguns casos somente para US complexas foi usado o formato BDD, e "o resultado foi espetacular" por terem ainda os critérios de aceitação descritos também. Muitos levantaram a questão do ganho de detalhes auxiliando no início do desenvolvimento e dos cenários de testes. Ainda foi destacado o fato de que quando foi preciso revisar uma história de usuário, aquela que se encontrava no formato BDD foi resolvida mais facilmente e, como consequência disso, obteve-se um ganho de tempo e qualidade.
- ✓ Dos 18 participantes, apenas 4 ainda não possuem todas as US em formato BDD, e de acordo com os membros do time, está sendo usando para as histórias de usuário mais complexas o que trouxe uma "diminuição drástica na quantidade de defeitos e reteste".

Neste contexto, foi possível identificar que todos os POs, arquitetos e *Scrum* Master tiveram a visão de que houve um ganho de qualidade, sendo que nem todos os desenvolvedores e analista de testes tiveram essa mesma visão no momento atual.

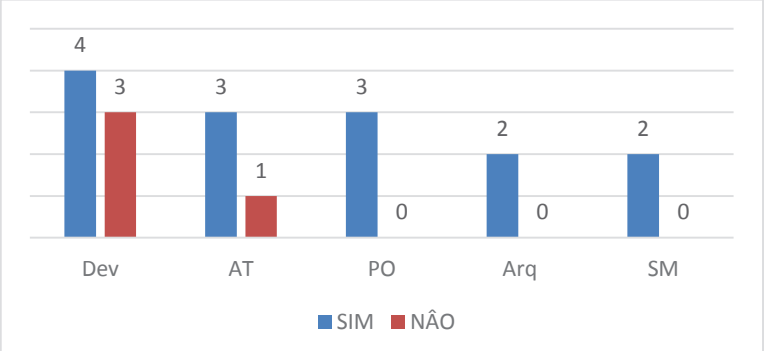

Figura 11 - Participantes do projeto referente a questão 4.

5 - Você acredita que com o uso do BDD todos da equipe tiveram um melhor entendimento do que era requisitado? Sim / Não. Por quê?

- ✓ Dos 18 participantes, 17 acreditam que os membros do time tiveram um melhor entendimento. Isto justifica-se pela percepção sobre se ter um padrão de escrita, fazendo com que o requisito ficasse claro, usando poucas palavras, o que auxilia principalmente nas histórias mais complexas. Além disso, alguns destacaram o fato de os cenários em BDD serem escritos e validados com o auxílio do time inteiro o que "auxilia enormemente na compreensão do que deve ser feito".
- ✓ Dos 18 participantes, 1 concorda parcialmente, pois em alguns casos nem todos do time conseguiram absorver no primeiro momento o entendimento do que era requisitado.

Apenas 1 (um) Scrum Master levantou a questão de que membros do time não tiveram um entendimento inicial do que era requisitado.

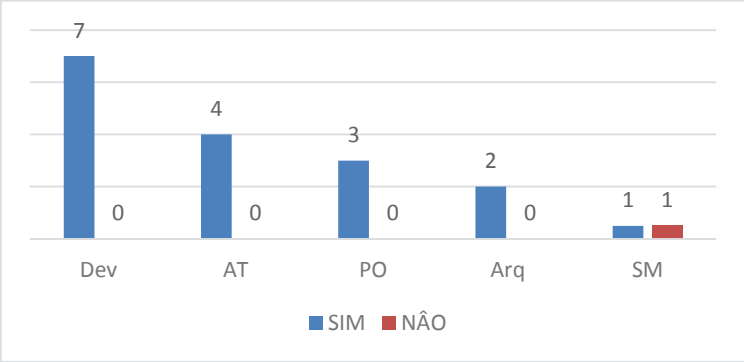

Figura 12 - Participantes do projeto referente a questão 5.

6 - Você acredita que com o uso do BDD melhorou a comunicação entre os membros da equipe? Sim / Não. Por quê?

- ✓ Dos 18 participantes, 13 acreditam que a comunicação melhorou por estimular a discussão em relação a detalhes como, por exemplo, os critérios de aceitação/cenários entre todos os membros do time.
- ✓ Por outro lado, dos 18 participantes, 5 não acreditam que o BDD possa ter influência nisso, ou ainda, que não reconheciam nenhum problema de comunicação no time.

Todos os analistas de testes tiveram a percepção de que houve uma melhora na comunicação, em contrapartida, outros membros não identificaram que houvesse problema de comunicação.

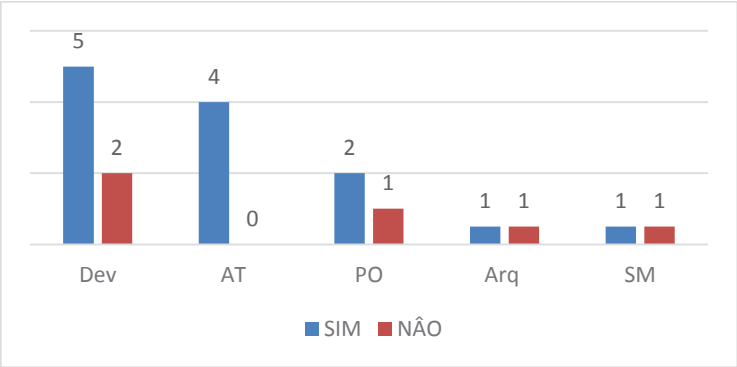

Figura 13 - Participantes do projeto referente a questão 6.

7 - Você acredita que com o uso do BDD o compartilhamento do conhecimento entre os membros da equipe (por ex: QA, DEV, PO, Scrum Master) melhorou? Sim / Não. Por quê?

- ✓ Dos 18 participantes, 14 afirmam que trouxe um ganho de sincronização das informações e estimulou a discussão sobre a história do Usuário, e ainda auxiliou para identificar alguns pontos falhos.
- ✓ Em contrapartida, dos 18 participantes, 3 não acreditam que o BDD possa ter influência nisto, mas pode colaborar de certa forma em relação à documentação.

Para todos os desenvolvedores e POs, foi perceptível a melhora do conhecimento entre os membros do time, mas para alguns analistas de teste, arquitetos e *scrum* master, o BDD não traria atuação neste contexto.

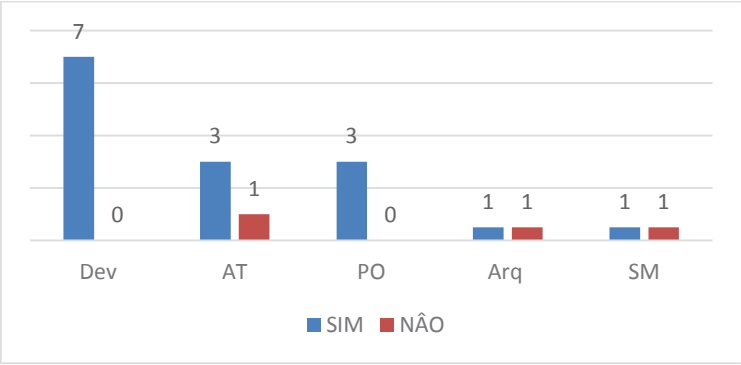

Figura 14 - Participantes do projeto referente a questão 7.

8 - Quais as vantagens e desvantagens que você vê com o uso do BDD?

- ✓ Todos os participantes, afirmam ter vantagens com o uso do BDD, pois há um ganho com a facilidade de identificar os cenários de testes, e estes podem ser transportados para os cenários de automação de testes. Alguns corroboram no sentido de ver grande vantagem no entendimento do time sobre os requisitos, facilitando para o desenvolvimento inicial e de testadores e ainda para o cliente,

pois existe uma linguagem comum entre todos. Além disso, se tem uma maior cobertura do código, menos *bugs*, e retrabalho, pois, todos do time se envolvem desde o começo.

- ✓ Dos 18 participantes, 6 não veem ou não identificaram desvantagens relacionadas ao uso do BDD.
- ✓ Dos 18 participantes, 9 apresentaram as seguintes desvantagens: alguns apontaram a questão da resistência a mudança por parte de algumas pessoas, dificultando a mudança no processo ou ainda impedindo que essa alteração pudesse ser concluída. Outro ponto levantado, é que na fase de planejamento há um esforço maior por parte da equipe e ainda que em *US* (histórias de usuário) simples seria colocado um esforço que não se justificaria. Além disso, um custo mais alto e manutenção do código de testes também foi mencionado.

Todos foram capazes de identificar vantagens, e nenhum dos POs apontou alguma desvantagem em relação ao BDD comparado as outras funções. Alguns desenvolvedores, Analistas de teste e arquitetos não identificaram desvantagens, e salientaram isso na resposta.

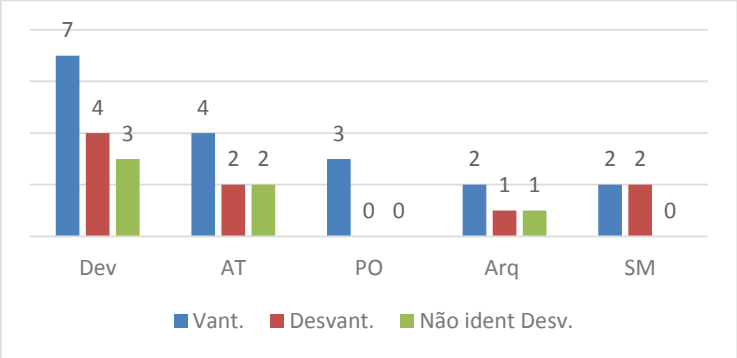

Figura 15 - Participantes do projeto referente a questão 8.

9 - Relacionado ao esforço do desenvolvimento, você acredita que houve menos retrabalho com o uso do BDD? Sim / Não. Por quê?

- ✓ Dos 18 participantes, 11 acreditam que sim pois, os cenários são bem detalhados, auxiliando também na área de testes. Pode-se destacar o fato de ter diminuído o reteste, quantidade de defeitos encontrados, e tempo despendido explicando os problemas para terceiros. E ainda, a compreensão do que é requisitado colabora para detectar falhas básicas no início do desenvolvimento.
- ✓ Dos 18 participantes, 5 no entanto, ainda não tem parâmetros para responder, ou ainda que o uso do BDD por si só, não garante o fim do retrabalho.

Todos os POs, tiveram a percepção de que houve menos retrabalho. Mas alguns desenvolvedores, analistas de teste, um arquiteto e um scrum master, compartilham da mesma percepção. Já os demais membros dos times não indicaram o quanto o BDD pode auxiliar neste ponto.

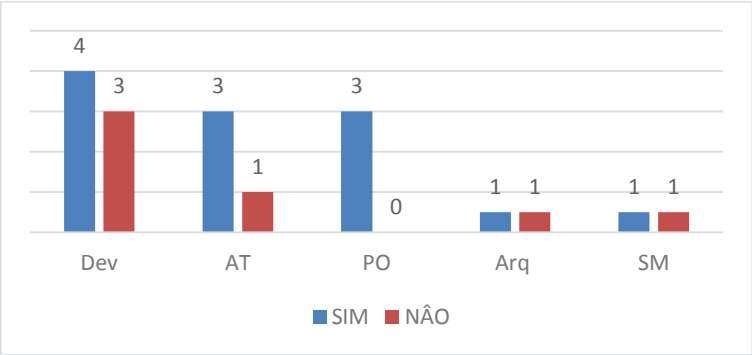

Figura 16 - Participantes do projeto referente a questão 9.

10 - De uma maneira geral, após a introdução e a aplicação do BDD no projeto, você acredita que foi possível transformar os requisitos em um produto que agrega valor ao cliente?

- ✓ Dos 18 participantes, 13 veem positivamente o uso BDD agregando valor, pois auxilia no entendimento dos cenários e possibilita maior compreensão das necessidades de negócio e de como o cliente espera interagir com o produto final.
- ✓ Sob outra perspectiva, dos 18 participantes, 4, não consideram o uso do BDD por si só o "responsável" por agregar valor.
- ✓ Dos 18 participantes, 1não respondeu a esta questão.

A maioria dos desenvolvedores, analistas e POs tem a visão de que através do BDD pode-se agregar valor ao cliente. Já alguns desenvolvedores, analistas, arquitetos e scrum máster, não reconhecem o BDD com essa função.

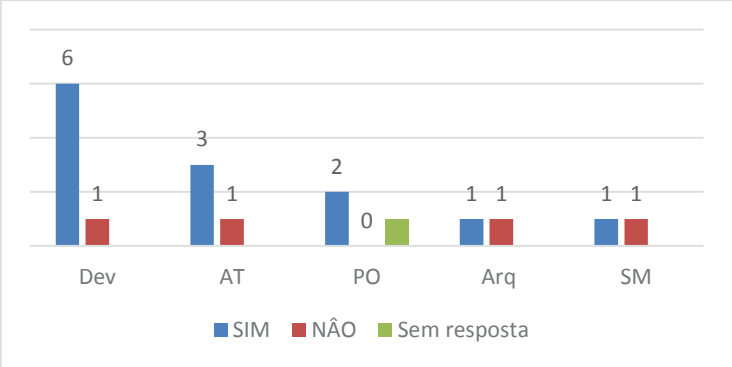

Figura 17 - Participantes do projeto referente a questão 10.

5. Conclusões

Ao término deste artigo observou-se que os objetivos gerais e específicos foram atingidos, uma vez que foi possível modificar o ambiente proposto pelo estudo e obter um bom resultado que foi capaz de apresentar argumentos que deram créditos ao uso do BDD como uma melhoria de processo.

É sabido que não existe uma "bala de prata" que possa sanar todas as lacunas ou pontos fracos em um desenvolvimento ágil de software, mas com o BDD alguns pontos foram nitidamente positivos em relação a isto.

Os aspectos da qualidade fazem parte do cotidiano do time, e após a implantação do BDD pode-se observar que isto proporcionou uma melhora no processo de desenvolvimento ágil para os projetos A e B. Essa percepção pode ser notada nas respostas obtidas, pois elas revelam que ocorreu uma melhora em vários sentidos relacionados ao entendimento das US.

Podemos destacar alguns pontos como:

- Todos conseguiram perceber valor no uso do BDD no projeto em que estavam trabalhando e também identificam vantagens no seu uso;

- Aumento da qualidade em função de ter os cenários mais detalhados e podendo identificar o que era esperado;

- A maioria dos membros dos times, consegue ter um entendimento do que é requisitado e se isso não acontecer, são sanadas dentro do próprio time com a ajuda do PO. Reuniões para tirar dúvidas quanto aos cenários são realizadas o que auxilia muito neste requisito.

Esse artigo apresenta argumentos para que a empresa possa futuramente introduzir o BDD em outros projetos, apresentando os seus benefícios relatados e percebidos pelos projetos que estão usando isto.

A contribuição deste estudo para a área de qualidade de software destaca-se pela melhoria de processo usando o BDD e de como seus resultados foram positivos para o projeto.

Em relação a limitação do estudo, na empresa escolhida para a realização do estudo, muitos outros projetos usam a metodologia ágil, sendo que alguns destes projetos encontram-se nos USA. Com isso, seria preciso um prazo maior para que o estudo fosse aplicado em todos os projetos e fosse verificado quais deles estariam dispostos a introduzir o BDD no seu processo de desenvolvimento e teste.

Por isto, o presente estudo foi realizado somente com projetos localizados no Brasil, e ainda nos projetos que aceitaram a introdução do BDD como sendo um meio de melhoria no processo. Dadas essas informações, o escopo da pesquisa foi reduzido e realizado de acordo com a conveniência e oportunidade para aplicar o estudo, carecendo de mais elementos para que seus resultados pudessem ser generalizados.

Referências

AGILE MANIFESTO. **Manifesto for Agile Software Development**. Disponível em <http://agilemanifesto.org/>. Acesso em: Junho de 2014.

AUDY, H Jorge. **Afinal, o que é o Manifesto e Métodos Ágeis?** 09 Jan. 2013. Disponível em:<http://www.baguete.com.br/colunistas/colunas/1173/jorge-horacio-audy/09/01/2013/afinal-o-que-e-o-manifesto-e-metodos-ageis> Acesso em: 6 Jun. 2015.

AUVRAY, Sebastien. **Receitas de Desenvolvimento Orientado a Estórias com Cucumber.** 09 Mar 2009. Disponível em:

<http://www.infoq.com/br/news/2009/03/bdd-with-cucumber> Acesso em: 5 abr. 2015.

BARTIÉ, Alexandre. **Garantia da Qualidade de Software. As melhores práticas de engenharia de software aplicadas à sua empresa.** 3. Ed. Rio de Janeiro: Campus, 2002.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 6 jun. 2015.

BORIA, J., Rubinstein, V. e Rubinstein A. "**A história da Tahini-Tahini – Melhoria de processos de software com métodos ágeis e modelo MPS"**. Ministério da Ciência, Tecnologia e Inovação, Secretaria de Política de Informática, Brasília. 2013.

COHN, Mike. **User Stories Applied: For Agile Software Development**. Boston. Ed. Addison-Wesley Professional, 2004.

CRISPIN, Lisa; GREGORY Janet. **Agile Testing: A Practical Guide for Testers and Agile Teams.** 1. Ed. Addison-Wesley Professional. Boston, 2009.

FLICK, Uwe. **Introdução à Pesquisa Qualitativa**. 3.ed. São Paulo: Artmed, 2009.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social.** 4.ed. São Paulo: Atlas, 1994.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

GODOY, Arilda Schmidt. **Introdução à Pesquisa Qualitativa e suas Possibilidades**. RAE – Revista de Administração de Empresas. São Paulo, v.35, n2, p.57-63. 1995

GRIEBLER, Fabricio. **Ciclo do Scrum**. 2011. Disponível em: <http://blog.fasagri.com.br/?p=125> Acesso em: 1 Jun 2015.

HELM, Rafael; WILDT, Daniel. **Histórias de Usuários**. **Por que e como escrever requisitos de forma Ágil?** 3.ed. historiasdeusuario.com.br, 2014.

JR, Elemar. **BDD na prática – parte 1 – Conceitos básicos e algum código**. 2012. Disponível em: < http://elemarjr.net/2012/04/11/bdd-na-prtica-parte-1-conceitos-bsicos-e-algum-cdigo/> Acesso em: 01 Jun 2015.

KOCH, Ricardo. L; OLIVEIRA, Leonardo. Rocha. **Riscos na Utilização de Métodos Ágeis na Gestão de Projetos de Software**. 2010. Disponível em: <http://www.convibra.com.br/upload/paper/adm/adm_1488.pdf> Acesso em 2 maio. 2015.

LEWIN, K. **Action research and minority problems.** Journal of Social Issues, Malden, v. 2, n. 4, p. 34-46, 1946.

MATTÉ, Marcio Angelo. **Testes de Software: Uma Abordagem da Atividade de Teste Software em Metodologias Ágeis Aplicando a Técnica Behavior Driven Development em um Estudo Experimental**. Medianeira: UTFPR (Universidade Tecnológica Federal do Paraná), 2011. (Monografia de Especialização em Engenharia de Software). Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/1111> Acesso em: 26 maio. 2015

MORAES, Roque. **Análise de conteúdo**. *Revista Educação*, Porto Alegre, v. 22, n. 37, p. 7-32, 1999.

MORESI, Eduardo. "**Metodologia de Pesquisa**". Universidade Católica de Brasília, 2003.

NORTH, Dan. **Introducing BDD.** 2003. Disponível em: <http://dannorth.net/introducing-bdd/> Acesso em: 06 abril. 2015.

OLIVEIRA, S. R. B. **"Conceitos Fundamentais de Qualidade de Software"** 2011. <Http://www.ufpa.br/srbo/disciplinas/cbcc_cbsi_mestrado_qualidade/aulas/aula02.pdf> Acesso em 18 jun. 2015.

OPPERMANN, Álvaro. **Porque os projetos falham?** 2014. Disponível em: <http://epocanegocios.globo.com/Inteligencia/noticia/2012/04/por-que-tantos-projetos-falham.html> Acesso em: 13 abr. 2015.

PHAM, Andrew; PHAM, Phuong-Van. **Scrum em Ação: Gerenciamento e desenvolvimento ágil de projetos de software.** Novatec, 2012.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. 7. ed. Porto Alegre: Artmed, 2011.

ROCHA, G. Fabio. **Scrum e BDD: o casamento perfeito.** 2013. Disponível em: <http://www.devmedia.com.br/scrum-e-bdd-o-casamento-perfeito/28174> Acesso em 12 abril 2015.

SMART, John F. (Foreword By Dan North) **BDD in Action: Behavior Driven Development for the whole software lifecycle.** Manning Publications Co. NY. 2015.

SOARES, I. **Desenvolvimento orientado por comportamento (BDD) - Um novo olhar sobre o TDD.** Java Magazinne, Rio de Janeiro, v. I, n. 91, 2011. Disponível em: <http://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd-artigo-java-magazine-91/21127> Acesso em 1 maio 2015.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

Wikipedia **Behavior Driven Development**. 2015. Disponível em: <http://pt.wikipedia.org/wiki/Behavior_Driven_Development> Acesso em 14 abr. 2015.

YIN, Robert K. **Estudo de Caso: Planejamento e métodos**. Bookman, trad. Daniel Grassi, ed. 2, Porto Alegre, 2001.

APÊNDICE A – Instrumento de Coleta de Dados

Avaliação do uso do BDD em desenvolvimento de Software

Pesquisa:

Este questionário faz parte de um trabalho de pesquisa para elaboração de um artigo do Curso de Pós Graduação em Qualidade de Software - Unisinos.

O objetivo deste questionário é, com base nas suas respostas, levantar infomações acerca do conhecimento, utlização e importância do BDD. (Behavior Driven Development).

Questões/Tempo: O questionário possui 10 questões e necessita, em média, de 10 a 15 minutos para ser respondido.

Para que este trabalho alcance o seu objetivo, sua colaboração é de suma importância. Suas respostas não serão consideradas certas ou erradas, o que importa é a sua opinião. Ressalto que o sigilo será mantido e suas respostas serão apresentadas genericamente.

Antecipo agradecimentos a seu apoio e disponibiliade para a realização deste trabalho.

*** 1. Há quanto tempo você trabalha com desenvolvimento de software Ágil? Em anos e meses.**

*** 2. Qual a sua função dentro da equipe? Dev / QA/ PO/ Arquiteto/ Scrum Master/Outros**

- ◯ Desenvolvedor
- ◯ Scrum Master
- ◯ Testador
- ◯ PO (Product Owner)
- ◯ Arquiteto de Software
- ◯ Outro

Outro (especifique)

*** 3. Você vê valor no uso do BDD no seu projeto?**

- ◯ Sim
- ◯ Não

Por quê?

*** 4. No seu projeto a US (user story) é escrita em formato de BDD? Se sim, tendo as US escritas no formato BDD, você acredita que isso trouxe um ganho de qualidade no desenvolvimento do software?**

- ◯ Sim
- ◯ Não

Por quê?

* 5. Você acredita que com o uso do BDD todos da equipe tiveram um melhor entendimento do que era requisitado?

- ◯ Sim
- ◯ Não

Por quê?

* 6. Você acredita que com o uso do BDD melhorou a comunicação entre os membros da equipe?

- ◯ Sim
- ◯ Não

Por quê?

* 7. Você acredita que com o uso do BDD o compartilhamento do conhecimento entre os membros da equipe (por ex: QA, DEV, PO, Scrum Master) melhorou?

- ◯ Sim
- ◯ Não

Por quê?

* 8. Quais as vantagens e desvantagens que você vê com o uso do BDD?

* 9. Relacionado ao esforço do desenvolvimento, você acredita que houve menos retrabalho com o uso do BDD?

- ◯ Sim
- ◯ Não

Por quê?

* 10. De uma maneira geral, após a introdução e a aplicação do BDD no projeto, você acredita que foi possível transformar os requisitos em um produto que agrega valor ao cliente?

Concluído

APÊNDICE B - Respostas dos colaboradores

3 - Você vê valor no uso do BDD no seu projeto? Sim / Não. Por quê?

SM1= "Sim. O uso do BDD ofereça mais clareza para todos integrantes do time sobre o que é necessários ser feito para um artefato ser entregue com sucesso. Com a adoção de BDD ocorreu uma aproximação maior entre QA e DEV, o que, em muitos projetos que participei, foi um desafio".

SM2= "Sim. Além do ganho de tempo, evita o retrabalho e consolida a qualidade do que está sendo entregue".

Arq1 = "Sim. Pois é uma maneira efetiva de associar documentação e cenários de teste em uma única linguagem".

Arq2 = "Sim. BDD ajuda descrever de forma clara os requisitos das User Stories, facilitando o entendimento das US pelo time, o mapeamento dos casos de testes e também funciona como documentação. Porém, BDD é verboso e consome tempo para ser feito e mantido, recomendo apenas para USs mais complexas".
PO1 = "Sim".
PO2 = "Sim. Deixa claro steps lógicos, critérios para aceitação e o que é esperado do produto".
PO3 = "Sim. O BDD ajuda a ilustrar as intenções do usuário e os resultados que ele pretende alcançar. Com isto, a interação usuário -> sistema fica mais aparente e o caso de uso menos transacional, o que possibilita ao time pensar em formas de atender o caso de uso plenamente".

AT1 = "Sim. Os requisitos, cenários de teste e critérios de aceitação se transformam em uma coisa só. O time todo acaba olhando para os requisitos e colaborando para melhora-los".

AT2 = "Sim. Pois os cenários de teste são definidos e pensados desde o início da sprint".

AT3 = "Sim. Normalmente encontramos erros antes de chegar no testador final (iria testar somente a UI) ".

AT4 = "Sim. O BDD deixa claro para todos os envolvidos no processo os requisitos. O que para uma User Story é basicamente os critérios de aceitação e por consequência é o guia para os testes".

Dev1 = "Sim. Com as definições de comportamento, o desenvolvedor pode prever muitos cenários enquanto desenvolve".

Dev2 = "Sim. O uso de BDD auxilia todos os envolvidos no processo de desenvolvimento a ter um melhor entendimento dos requisitos de cada US".

Dev3 = "Sim. BDD ajuda a deixar claro os critérios de aceitação de uma história, especialmente quando são requisitos complexos. Por outro lado, o BDD quando bem feito também ajuda a garantir que requisitos antigos ainda estão sendo seguidos sem a necessidade de sprints inteiras de testes de regressão. Garantindo assim mais produtividade ao time".

Dev4 = "Sim. O uso de BDD garante uma maior consistência no desenvolvimento das funcionalidades solicitadas, fazendo com que seja bastante restringido a quantidade de tarefas voltando para a etapa de desenvolvimento por falhas nos critérios de aceitação".

Dev5 = "Sim. Facil entendimento pelo Dev e Stakeholder. Minimiza o esforço de criar Unit Tests".

Dev6 = "Sim".

Dev7 = "Sim. Pois é uma linguagem comum entre Business/Dev/QA".

4 - No seu projeto a US (*user story*) é escrita em formato de BDD? Se sim, tendo as US escritas no formato BDD, você acredita que isso trouxe um ganho de qualidade no desenvolvimento do software?

SM1= “Sim. Escrever as US em formato de BDD uniformizou a maneira de se escrever a US e a interpretação de todos sobre aquela US se tornou mais parecida”.

SM2= “Sim. Porque trouxe mais certeza aos critérios de aceitação das US's, além de trazer qualidade desde o início do desenvolvimento”.

Arq1 = “Sim. Sim, pois as US são, necessariamente, bem detalhadas”.

Arq2 = “Sim. Usamos por enquanto em apenas uma US, bem complexa, e tem atendido satisfatoriamente. É fácil entender o requisito, comunicar como ele funciona para outros times, e criar cenários de teste”.
PO1 = “Sim. Ainda não é possível dizer. O time tem pouco tempo trabalhando no produto e o uso de BDD é recente. Acredito ser necessário mais tempo para entender o impacto na qualidade do desenvolvimento”.
PO2 = “Sim. Já foi preciso revisar uma história com BDD e uma sem BDD. A com BDD foi resolvida mais facilmente”.
PO3 = “Sim. A US é escrita utilizando a frasiologia do BDD com foco em scripts de teste automatizado. Existe um ganho de compreensão tanto do time de Dev quanto QA ao aplicar verbos do BDD”.

AT1 = “Sim. Comecei no agile development já com BDD antes utilizava Waterfall, relacionando com o waterfall o BDD trouxe um ganho muito grande”.

AT2 = “Sim. Pois os cenários de teste são definidos e pensados desde o início da sprint, agregando qualidade devido aos questionamentos a estes cenários”.

AT3 = “Não. Estão começando a utilizar BDD. (só usamos em algumas US complexas) ”.

AT4 = “Sim. Não são todas, utilizamos o BDD quando a US possui muitos pontos de verificação, muitas regras de negócio e comunicações com times externos que precisam ser feitas”.

Dev1 = “Sim. Com a US escrita em BDD, muitas das dúvidas do desenvolvedor já estão respondidas, assim o desenvolvedor pode focar nas necessidades de código”.

Dev2 = “Sim. Utilizando BDD, além de se preocupar com a validação específica do componente que está em desenvolvimento também há uma visão geral de como o componente deve se comportar inserido no contexto do sistema. Certamente isso refletiu positivamente na qualidade”.

Dev3 = “Não. Nossas USs não são em formato BDD, mas temos um acordo dentro do time que histórias dentro de determinado tema deve incluir uma tarefa de criação dos BDDs e isso sim trouxe uma diminuição drástica na quantidade de defeitos e reteste realizado para determinados tipos de features”.

Dev4 = “Sim. Não todas, porém já utilizamos essa metodologia em algumas histórias. O resultado foi espetacular! Se as histórias forem todas construídas com seus critérios de aceitação em cima de padrões utilizados em ferramentas de BDD, os resultados do desenvolvimento seriam muito mais consistentes, uma vez que os critérios de aceitação serão literalmente aplicados sobre a codificação para que seja feita, através de processos automatizados, a validação das funcionalidades desenvolvidas. Como consequência disso, as etapas de testes não seriam tão sobrecarregadas de responsabilidade em cima da consolidação dessas funcionalidades, além do benefício especificado na Questão 3”.

Dev5 = “Não. Na verdade, algumas foram, no entanto, a minoria. Depois de definido o cenário e steps principal, era possível com pouca modificação, testar outros cenários”.

Dev6 = “Sim”.

Dev7 = “Não”.

5 - Você acredita que com o uso do BDD todos da equipe tiveram um melhor entendimento do que era requisitado? Sim / Não. Por quê?

SM1= “Não. Eu diria que, parcialmente sim. Alguns membros da equipe apresentaram dificuldades de entendimento sobre o mínimo necessário daquela user stories necessitando o detalhamento ao longo do período de desenvolvimento”.

SM2= “Sim. Porque deixa claro, em poucas palavras, o que é requisitado, sem sombra de dúvidas”.

Arq1 = “Sim. Porque as user stories são bem definidas e seguem um formato padrão, que facilita sua interpretação”.

Arq2 = “Sim. Isso é verdade apenas para US complexas, US simples acabam tendo um overhead grande e não trazendo valor suficiente para justificar o esforço”.
PO1 = “Sim. Como os cenários BDD são escritos e validados com auxílio do time inteiro, no mínimo há entendimento comum dos cenários discutidos”.
PO2 = “Sim”.
PO3 = “Sim. Os cenários ficaram mais claros e mais previsíveis para o time”.

AT1 = “Sim. Sim, pois todos precisam revisar os cenários e dar opiniões”.

AT2 = “Sim. Pois os cenários de teste são definidos e pensados desde o início da sprint, agregando qualidade devido aos questionamentos a estes cenários”.

AT3 = “Sim. Por que o critério de aceitação ficou mais claro, e novos cenários já foram encontrados/discutidos logo na elaboração do BDD, assim o time todo ficou conhecedor do que se trata a US realmente”.

AT4 = “Sim. Pelo simples fato de que fica claro para todos os envolvidos no processo o que precisa ser feito”.

Dev1 = “Sim. Sim, pois com o BDD escrito em uma linguagem clara todos sabem o comportamento dos componentes desenvolvidos”.

Dev2 = “Sim. Sim, o BDD proporciona a todos os envolvidos no projeto definições claras de como o produto deve se comportar. Claramente os requisitos são definidos e revisados de uma forma que garante o entendimento a todos envolvidos”.

Dev3 = “Sim. Melhor descrição dos critérios e detalhes de casa US”.

Dev4 = “Sim. A utilização do BDD deixa bem mais claro e explícito os objetivos da história, o que auxilia enormemente na compreensão do que deve ser feito e na garantia de que se está implementando o requisitado”.

Dev5 = “Sim”.

Dev6 = “Sim”.

Dev7 = “Sim”.

6 - Você acredita que com o uso do BDD melhorou a comunicação entre os membros da equipe? Sim / Não. Por quê?

SM1= “Não. O uso de BDD, exclusivamente, não ajudou na melhoria da comunicação. Em alguns casos, tivemos cenários que não foram implementados e esse erro foi aparecer somente no final da fase de testes, acredito que se o BDD tive auxiliado na comunicação esse erro não demoraria tanto para aparecer”.

SM2= “Sim. Porque estimula o time a sanar dúvidas entre suas diferentes especialidades (QA, DEV, PO, etc...) ”.

Arq1 = “Não. Não acho que comunicação fosse um problema entre os membros do time”.

Arq2 = “Sim”.
PO1 = Sim. Uma forma simples de compartilhar e estimular a discussão em relação a detalhes”.
PO2 = “Não”.

PO3 = “Sim. Existe um entendimento comum do que pretende ser atingido”.

AT1 = “Sim. BDD é uma ferramenta colaborativa, todos têm que ler os cenários BDD”.

AT2 =“Sim. Sim, pois são discutidas as definições escritas em formato BDD, e remodeladas quando necessário”.

AT3 = “Sim. Levanta discussões sobre os cenários a ser testados”.

AT4 = “Sim. Ao se escrever uma US em formato BDD todos membros do time participam na escrita e com isto se revisa todos os pontos desta US”.

Dev1 = “Sim. Sim, pois todos possuem objetivos alinhados, já que todos participam das definições”.

Dev2 = “Não. Não acredito que a aplicação de BDD tenha influenciado na comunicação do time”.

Dev3 = “Não. Acho que não teve influência”.

Dev4 = “Sim. Auxilia na comunicação desde que os critérios de aceitação sejam construídos com toda a equipe reunidamente. Afinal, ao trabalhar esses critérios de forma coletiva resultará em uma compreensão muito mais ampla entre os membros da equipe e fará com que todos se compreendam na hora de discutir sobre as funcionalidades em questão”.

Dev5 = “Sim”.

Dev6 = “Sim”.

Dev7 = “Sim”.

7 - Você acredita que com o uso do BDD o compartilhamento do conhecimento entre os membros da equipe (por ex: QA, DEV, PO, Scrum Master) melhorou? Sim / Não. Por quê?

SM1= “Não. O uso do BDD somente não auxiliou no compartilhamento de conhecimento”.

SM2= “Sim”.

Arq1 = “Não. Não entendo como uma ferramenta que sirva para melhorar o compartilhamento do conhecimento, mas é certamente uma ferramenta que melhora a centralização da documentação (e, portanto, uma parte do conhecimento) de forma centralizada e de fácil acesso”.

Arq2 = “Sim”.
PO1 = “Sim. Não posso dizer se melhorou porque é um time novo introduzindo BDD, mas estimulou a discussão sobre as necessidades do produto”.
PO2 = “Sim. Documentação com formatação específica ajuda na leitura e compreensão”.
PO3 = “Sim. Existe um entendimento comum do que pretende ser atingido”.

AT1 = “Sim. Comparando com o waterfall, a comunicação melhorou, pois, todos do time tem que se envolver na criação dos cenários”.

AT2 = “Sim. Sim, pois são discutidas as definições escritas em formato BDD, e remodeladas quando necessário”.

AT3 = “Sim”.

AT4 = “Não. Não notei diferença no uso do BDD para o compartilhamento de conhecimento”.

Dev1 = “Sim. Sim, todos os membros do time podem colaborar e também questionar o BDD, possíveis erros podem ser identificados antes do desenvolvimento”.

Dev2 = “Sim. Sim, acredito que há mais passagem de conhecimento. Principalmente de QA para DEV, onde o desenvolvedor tem mais contato com os cenários que serão executados para cada US, e com o processo de testes como um todo”.

Dev3 = ‘Sim. Não só melhorou o compartilhamento interno como é mais absurdamente mais facil explicar a parceiros externos quais as mudanças feitas apresentando apenas os BDDs”.

Dev4 = "Sim. Com toda certeza. Seguindo a atividade descrita na Questão 6, o uso do BDD resulta em um grande benefício para DEVs, QAs e POs: sincronia. Ao desenvolver os critérios de aceitação de forma coletiva, fica muito claro para todos os DEVs e QAs o que o PO está solicitando, da mesma forma que o PO garante que tudo o que é necessário está sendo explicitado".

Dev5 = "Sim".

Dev6 = "Sim".

Dev7 = "Sim".

8 - Quais as vantagens e desvantagens que você vê com o uso do BDD?

SM1= "Vantagens: mais cobertura do código, diminuir o tempo de identificação e correção de bugs e servir como documentação. Desvantagens: custo mais alto e manutenção do código de testes".

SM2= "Vantagem: mais garantia de entendimento do que está sendo requisitado, além da redução do retrabalho. Desvantagem: maior esforço por parte da equipe na fase de planejamento".

Arq1 = "Vantagens: Grande parte da documentação funcional e cenários de teste são cobertos por ele. Não vejo desvantagens".

Arq2 = "Vantagens: Documentação, facilidade para mapeamento de cenários de teste, entendimento do time. Desvantagem: US simples acabam tendo um overhead grande que não justifica o esforço, já que o benefício acaba não sendo tão grande".

PO1 = "Time inteiro envolvido para descrever os cenários. Facilidade para identificar novos cenários por qualquer um no time. Facilidade para transpor os cenários BDD em cenários de automação de testes".

PO2 = "Citadas acima".

PO3 = 'Os cenários se tornam menos transacionais e mais interativos".

AT1 = "Vantagens: Comunicação; facilidade na escrita de testes e automação; Melhor entendimento no que deve ser desenvolvido; Desvantagens: Se feita só por alguns membros do time e não todos podem ter problemas de entendimento".

AT2 = "Vantagens: - Escrita em formato definido desde o início da *sprint*; - Discussão fomentada por estas definições no início da *sprint*; - Replanejamento de casos não pensados desde o início da *sprint*. Desvantagens: - Não vi até o momento".

AT3 = "Vantagens: levantamento de cenários já no início do desenvolvimento".

AT4 = "Vantagem está na agilidade de ter a própria User Story automatizada só pelo formato da escrita dela. A desvantagem está na mudança de pensamento para a implantação deste modelo de trabalho, trabalho em uma empresa que tem POs, DEVs, QAs tradicionais e fazer estas pessoas mudarem a maneira de trabalhar ou pensar seu trabalho é algo complicado".

Dev1 = "Vantagens são as possibilidades de maior integração do time e na clareza das definições tanto para o desenvolvedor quanto para os testadores. Desvantagem eu acredito que tenho só no início, em que as vezes pode existir um pouco de resistência para adotar o BDD no ciclo de desenvolvimento".

Dev2 = "Praticamente todos os benefícios aplicáveis a TDD podem ser citados para o BDD. O que diferencia o TDD do BDD é a colaboração dos analistas de negócio (nesse caso o PO) com os *stakeholders* do cliente além do time de DEV e QA. E também uma definição de linguagem comum entre todos os envolvidos, de fácil entendimento tanto para o cliente quanto para o desenvolvedor".

Dev3 = "Por outro lado, como vantagens, o BDD possibilita melhora nos níveis de qualidade, diminuição da manutenção e maior confiança no produto entregue. Como desvantagem principal

acho que é a curva de aprendizado e o esforço inicial para mudar a forma de pensamento no processo de desenvolvimento".

Dev4 = "As maiores vantagens foram descritas nas perguntas anteriores. Ao meu ver, as desvantagens do BDD são ínfimas diante dos benefícios, pois o único tempo extra despendido é para se adaptar com as ferramentas de BDD e na manutenção dos métodos de teste. Contudo, esse tempo se dilui no tempo que será economizado tendo que refazer códigos diante de interpretações errôneas dos critérios de aceitação e bugs facilmente detectáveis com as ferramentas de BDD. Quanto ao tempo utilizado para construir os critérios de aceitação, é um tempo diluído nas Groomigs e Plannings, pois é um trabalho feito no lugar da metodologia padrão de construir as User Stories".

Dev5 = "Descrições de requisitos se tornam mais simples e claras. Fácil de estender para criar novos testes. Ajuda a criar uma linguagem comum entre desenvolvedores e clientes".

Dev6 = "Vantagens: User Story é escrita de uma forma que facilita o desenvolvimento e os testes gerados, garantindo que os mesmos serão consistentes com o valor requisitado. Desvantagens: Entregáveis demoram mais tempo para chegar ao cliente final".

Dev7 = "Vantagens é a facilidade de entendimento. Dificuldades é a dificuldade e custo para a escrita de maneira correta".

9 - Relacionado ao esforço do desenvolvimento, você acredita que houve menos retrabalho com o uso do BDD? Sim / Não. Por quê?

SM1= "Não. Pelo tempo que utilizo BDD ainda não pude observar a diminuição do retrabalho".

SM2= "Sim. Sim, principalmente na área de teste e na correção de erros".

Arq1 = "Sim. Sim, especialmente quando os cenários das US são bem detalhados".

Arq2 = "Não. BDD sim adiciona um esforço extra, utilizar para US simples não vale a pena, é preferível utilizar nas US que são complexas".
PO1 = "Sim. Como PO, não posso falar em termos de desenvolvimento. O que observo é um número bastante baixo de issues durante o desenvolvimento. Acredito que isso possa ser um bom indicativo que há menos retrabalho".
PO2 = "Sim. Adicionado mais um step no desenvolvimento, porém o retrabalho é reduzido".
PO3 = "Sim. Sim, em partes. A clareza sobre o cenário não implica que uma US saia 100% correta a primeira vez. Existem variáveis relacionadas a usabilidade que podem ser revisitadas a qualquer momento, mesmo que o cenário desejado já tenha sido atingido".

AT1 = "Sim. Sem o BDD seriam necessários documentos de requisitos, de critérios de aceitação, casos de testes".

AT2 = "Sim. Sim, pois são discutidas as definições escritas em formato BDD, e remodeladas quando necessário".

AT3 = "Sim".

AT4 = "Não. Não tenho parâmetros para responder esta pergunta".

Dev1 = "Sim. Sim, pois os desenvolvedores agora possuem mais conhecimento sobre os comportamentos dos componentes".

Dev2 = "Não. Não acredito. Penso que ainda teremos retrabalho com o projeto. O uso do BDD por si só não garante o fim do retrabalho. No nosso caso penso que os requisitos de negócio não foram levantados de forma adequada e foram definidos como consenso dentro do time, sem ter contato direto com o time de negócios do cliente".

Dev3 = "Sim. Sem sombra de dúvidas. Diminuiu o reteste, a quantidade de defeitos encontrados e, posteriormente, o tempo gasto explicando/encontrando problemas encontrados por terceiros".

Dev4 = “Sim. Vide as respostas das questões anteriores. Resumindo: real compreensão do que é requisitado e falhas básicas que são detectadas antes de chegar na etapa de testes”.

Dev5 = “Não. Não acredito que o BDD tenha como premissa diminuir o retrabalho, mas sim ajudar na qualidade do que é entregue”.

Dev6 = “Não”.

Dev7 = “Sim. Principalmente nos casos em que completamos com o Cucumber para gerar testes automatizados”.

10 - De uma maneira geral, após a introdução e a aplicação do BDD no projeto, você acredita que foi possível transformar os requisitos em um produto que agrega valor ao cliente?

SM1= “Não credito ao BDD transformar o requisito em um produto que agrega valor. Se uma User Story é bem planejada e bem definida ela vai agregar valor com ou sem BDD”.

SM2= “Sim, pois melhora muito a compreensão da real demanda do cliente, o que possibilita uma solução mais eficiente para os problemas”.

Arq1 = “Não vejo o BDD como responsável por isso. Acredito que o BDD auxilia na utilização de uma linguagem comum, que pode ser compreendida por todos, desde PO, a desenvolvedor, ao cliente - e agrega valor na compreensão destes requisitos”.

Arq2 = “Sim, a documentação gerada auxiliou times externos a entenderem melhor o funcionamento da funcionalidade abstraindo termos técnicos”.

PO1 = “Sem resposta”.

PO2 = “Sim, não especificamente por causa do BDD, ele somente dá suporte para agregar valor”.

PO3 = “Certamente sim. O BDD auxilia no entendimento dos casos de uso e de como o usuário espera interagir com o produto. Isto faz com que deixemos de lado cenários puramente transacionais e passemos a trabalhar com interações entre usuário e sistema. Seguindo este formato, temos melhor cobertura dos casos de uso e por consequência mais valor para o usuário final”.

AT1 = “Sim após utilizar o BDD em um projeto, o produto foi entregue com tudo que foi requisitado”.

AT2 = “Sim. Sim, com o auxílio da escrita das user stories baseadas em comportamento”.

AT3 = “Acho que melhorou um pouco pois podemos forçar o desenvolvedor a pensar (desde o início) nos casos/cenários que o cliente irá executar”.

AT4 = “Não”.

Dev1 = “Sim, com BDD é possível também criar testes automatizados que podem ser executados com mais frequência e diminuindo ainda mais os possíveis problemas”.

Dev2 = “Acredito que não. Como mencionado anteriormente para que o BDD (ou qualquer outra metodologia de testes ou desenvolvimento) trazer os resultados esperados as definições de requisitos devem ser bem-feitas. As metodologias ágeis, no geral pregam mais contato com o cliente e entregas mais rápidas para prever e atuar rapidamente as mudanças de escopo que possam surgir. Mas o uso do BDD por si só não vai fazer com que o produto seja o que o cliente espera se os cenários de comportamento do sistema não refletirem o que o cliente espera”.

Dev3 = “Acho que os requisitos quando colocados lado a lado já são produtos que agregam valor ao cliente, então o papel do BDD nisso é facilitar o entendimento destes requisitos aos times de desenvolvimento possibilitando maior compreensão das necessidades de negócio e empodera os times para sugerir melhorias”.

Dev4 = “Sim, pois o BDD garante que as funcionalidades, por mais complexas que possam ser, sejam implementadas de forma consistente com que foi solicitado, sem que aja transtornos com reparação de defeitos, acarretando demora de entrega, aumento de custos, entre outros”.

Dev5 = “Sim, mas como utilizamos o BDD como piloto em apenas algumas US, o ganho não foi tão perceptível. Acredito que o ganho maior seria a longo prazo, através da cobertura do código do sistema que iria ajudar nos *refactories*, bem como na identificação mais fáceis de erros introduzidos por modificações no sistema. Basicamente estes são fatores na verdade herdados de testes unitários de código”.

Dev6 = “Acredito, devido à alguns fatores como a consistência do produto final e a facilidade do entendimento de regras de negócio, etc”.

Dev7 = “Sim”.