SEMANA ACADÊMICA
REVISTA CIENTÍFICA
ISSN 2236-6717

APLICAÇÃO DA TÉCNICA BDD COMO APERFEIÇOAMENTO EM UM PROJETO ÁGIL (SCRUM)

[ver artigo online]

Autor [1] Fabricio Micheletti Rodrigues
Autor [2] Ma. Renata Mirella Farina

RESUMO

Este artigo tem por finalidade apresentar de forma objetiva a introdução da técnica de Desenvolvimento Dirigido por Comportamento, oriundo do termo inglês *Behavior Driven Development* (BDD) com o intuito de aprimorar e apressurar um projeto ágil, utilizando o framework de gerenciamento de projetos Scrum. De forma geral, elucidará primordialmente os processos e avanços alcançados no decorrer dos anos no que é pertinente ao desenvolvimento de software ágil até a evolução tecnológica ter condições de elaborar técnicas como a do BDD, bem como demonstrar os motivos que ocasionaram a sua necessidade de utilização. Para verificação do comportamento de referida técnica foram realizadas coletas de dados, observações e avaliação dos conteúdos submetidos à supracitada técnica. O conjunto de análises realizadas demonstraram positivamente os impactos obtidos com a utilização do BDD no desenvolvimento de software ágil e possibilitou avaliar, além da forma teórica, a prática da implicação da aplicação do BDD no cotidiano dos profissionais envolvidos num determinado projeto.

Palavras-chave: Desenvolvimento Ágil, Scrum, BDD.

APPLICATION OF BDD TECHNIQUE AS IMPROVEMENT IN AN AGILE PROJECT (SCRUM)

ABSTRACT

This article aims to objectively present the introduction of the Behavior Driven Development technique, derived from the English term Behavior Driven Deve-lopment (BDD) in order to improve and speed up an agile project, using the management framework Scrum projects. In general, it will primarily elucidate the processes and advances achieved over the years in what is pertinent to the development of agile software until technological evolution is able to develop techniques such as BDD, as well as demonstrate the reasons that led to its need to use. To verify the behavior of this technique, data collection, observations and evaluation of the contents submitted to the aforementioned technique were carried out. The set of analyzes carried out positively demonstrated the impacts obtained with the use of BDD in the development of agile software and made it possible to evaluate, in addition to the theoretical form, the practical implications of applying BDD in the daily lives of professionals involved in a given project.

Key-words: Agile Development, Scrum, BDD.

1 Graduando do Curso de Engenharia da Computação da Universidade de Araraquara- UNIARA. São Paulo, fabricio.2505@hotmail.com

2 Docente do curso de Engenharia da Computação, mestre em Engenharia de Produção, graduada em Administração de Empresas, Estudos Sociais, Geografia e Análise de Sistemas, Universidade de Araraquara- UNIARA. São Paulo, rmfarina@uniara.edu.br

http://dx.doi.org/10.35265/2236-6717-209-9305
FORTALEZA-CE. EDIÇÃO 209. V.9. ANO 2021.

INTRODUÇÃO

Assim como frequentemente é observado, na grande maioria dos âmbitos de relacionamentos pessoais, a falha de comunicação e/ou entendimento nas mais vastas circunstâncias cotidianas, com os profissionais envolvidos nos processos de desenvolvimento ágil de software não é diferente. Não é raro que, mesmo após um projeto ter sido submetido à todas as etapas para sua conclusão e, consequentemente, atendimento da necessidade previamente relatada pelo cliente ao procurar por uma empresa de tecnologia, ao final sejam identificadas falhas de comunicação/entendimento, que acarretam maior demanda de tempo e trabalho a fim de, somente então, satisfazer à carência de referido cliente.

Segundo Koscianski (2006) a qualidade do software depende em grande parte dos requisitos. Caso os requisitos venham a ser mal definidos ou mal escritos, isso dará origem a um produto que não foi solicitado pelo cliente. Todo requisito que venha sofrer um desvio ao que foi estabelecido inicialmente pode ocasionar em prejuízos diversos, tendo oscilações de acordo com a necessidade e utilização de cada sistema.

Ainda que diariamente sejamos surpreendidos e, por vezes, atropelados pela sede incessante da evolução tecnológica, o processo pertinente ao desenvolvimento de software é complexo e, segundo Cohn (2004, p3) "Requisito de software é um problema de comunicação".

De acordo com Oppermann (2014), após um estudo realizado pelo Network de firmas independentes PwC, antigamente conhecida como PriceWaterhouseCooper, com mais de 200 empresas localizadas em diversos países, identificou-se que somente 2,5% delas obtiveram sucesso em relação aos seus projetos nos últimos anos, onde estas empresas focam demais nas práticas e processos do que nas pessoas envolvidas e na clara comunicação.

É fato que, assim como empresas dos mais diversos ramos de atuação que investem em técnicas de comunicação para que haja clara e única compreensão de todos os setores envolvidos tendem a obter resultados de forma mais ágil e satisfatória, nas empresas de tecnologia, é visível o valor agregado com a aplicação de uma boa técnica em relação aos requisitos, que podemos considerar fator primordial para que os profissionais entendam o que realmente foi solicitado pelo cliente e assim, seja evitado que o sistema entregue divirja-se do conceito principal.

De forma análoga e cotidiana, podemos entender o BDD, por exemplo, como a cura nas interferências telefônicas. Se, durante uma ligação telefônica, por qualquer motivo haja interferência de sinal, a mensagem emitida será compreendida de forma distorcida pelo receptor e ocasionará conflitos na intercomunicação, que somente serão solucionados na repetição do diálogo. A aplicação do BDD, portanto, visa resolver as interpretações errôneas para que não haja a necessidade de retrabalho, sem qualquer desvio, com um único objetivo: sanar a dor do cliente.

O presente artigo, portanto, tem como objetivo geral implementar um aperfeiçoamento no processo de desenvolvimento ágil (que utiliza a metodologia Scrum) utilizando a técnica de BDD, proporcionando um melhor entendimento das solicitações realizadas pelo cliente da empresa "ABC", que é uma multinacional de consultoria que oferece soluções de negócios, estratégia, desenvolvimento e manutenção de aplicações tecnológicas. Atuando nos setores de telecomunicações, industriais, de energia e saúde. Contendo mais de 10.000 profissionais localizados em escritórios e centros em mais de 14 países. Tendo sua sede no Brasil localizada na cidade de Uberlândia-MG.

1 MANIFESTO ÁGIL

Com o passar dos anos a humanidade tem buscado obter soluções a cada dia mais rápidas. Estamos imersos numa corrida incansável por evolução e mudanças ágeis e os resultados precisam ser quase imediatos.

O imediatismo da atualidade faz com que as empresas, consequentemente, enxerguem a necessidade de possuir respostas breves para as novas tendências e novos mercados que surgem. Sommerville (2011, p38) afirma que:

> “O desenvolvimento e entrega rápidos são, portanto, o requisito mais crítico para o desenvolvimento de sistemas de software, na verdade, muitas empresas estão dispostas a trocar a qualidade e o compromisso com requisitos do software por uma implantação mais rápida do software de que necessitam. ”

O que nos remete ao "Manifesto Ágil", criado em 2001, por Kent Beck e mais dezesseis renomados profissionais da área.

1.1 Princípios primordiais do Manifesto Ágil

- Nossa primeira e grande prioridade é a satisfação do cliente com entrega adiantada e contínua de software com valor;
- Aceitar mudanças e se adequar a elas para que o cliente venha a obter vantagens competitivas no mercado, mesmo que no fim do projeto;
- Frequência na entrega de software funcionando, em poucas semanas e meses, sempre aderindo ao menor tempo;
- Desenvolvedores e *stakeholders* devem trabalhar em conjunto diariamente no projeto;
- Manter indivíduas motivados com projetos que são construídos em volta de um ambiente e suporte adequados e confiar o trabalho a eles;
- Transmitir informações de forma mais clara e eficaz dentro de uma equipe de é por meio de conversa cara a cara;
- A medida principal de progresso é software funcionando;
- Usuários, desenvolvedores e patrocinadores devem manter um ritmo constante indefinidamente, pois os processos ágeis promovem desenvolvimento sustentável;
- Bom design e atenção a excelência técnica, promovem o aumento da agilidade;
- Potencializar a quantidade de trabalho\retrabalho não realizado é uma arte essencial;
- Times organizáveis detêm de designs, requisitos e arquiteturas melhores;
- Os comportamentos são refinados e ajustados em reflexões da equipe para se tornar mais eficaz ao longo do tempo.

Além das esferas tecnológicas, o empenho humano é crucial para que tudo flua de forma contínua.

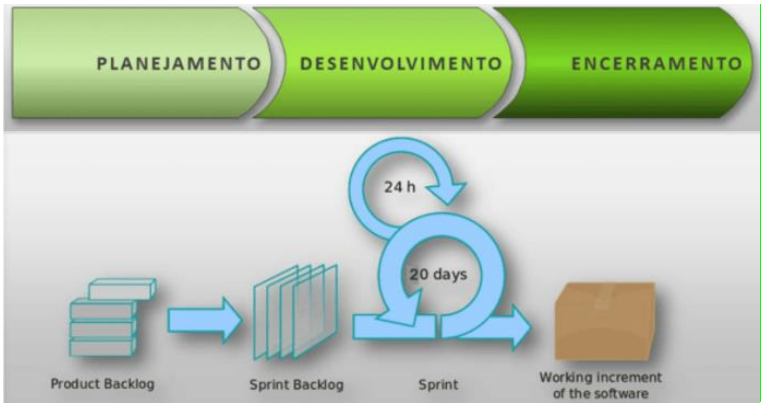
2 DESENVOLVIMENTO ÁGIL DE SOFTWARE

De forma, inclusive maçante, é impossível entendermos as características da técnica de BDD sem antes pincelarmos sobre o que, de fato, é o desenvolvimento ágil de software.

Suscintamente, podemos entendê-lo como o conjunto de comportamentos, processos, práticas e ferramentas utilizadas na criação de produtos e a sua disponibilização para o cliente final.

O desenvolvimento ágil de software ou método ágil tem o intuito de minimizar os riscos existentes em curtos períodos e para tal, existem inúmeros frameworks de processos.

Figura 1 – Fases do desenvolvimento segundo a MDS



Fonte: Suframa

3 SCRUM

Com o avanço da tecnologia surgiu a necessidade de serem desenvolvidos métodos facilitadores e ágeis para serem aplicados durante o desenvolvimento de projetos.

O supracitado imediatismo da atualidade, acrescido da evolução tecnológica, impulsionaram o uso de diversos frameworks, dentro eles e nos dias de hoje, o mais usual, podemos citar o Scrum, que, por meio de práticas que oferecem alternativas adaptáveis, faz uso da criatividade para solucionar os casos, organiza os projetos por Sprint (etapas), planeja a execução de cada Sprint, orienta o trabalho de cada profissional envolvido e por consequência garante melhor eficácia, é um atenuante de problemas complexos.

De acordo com Pham (2012), a metodologia Scrum surgiu em um artigo de Hirotaka Takeuchi e Ikujiro Nonaka publicado na Harvard Business Review de 1986, denominado "*The new product development game*" (O novo jogo do desenvolvimento de novos produtos).

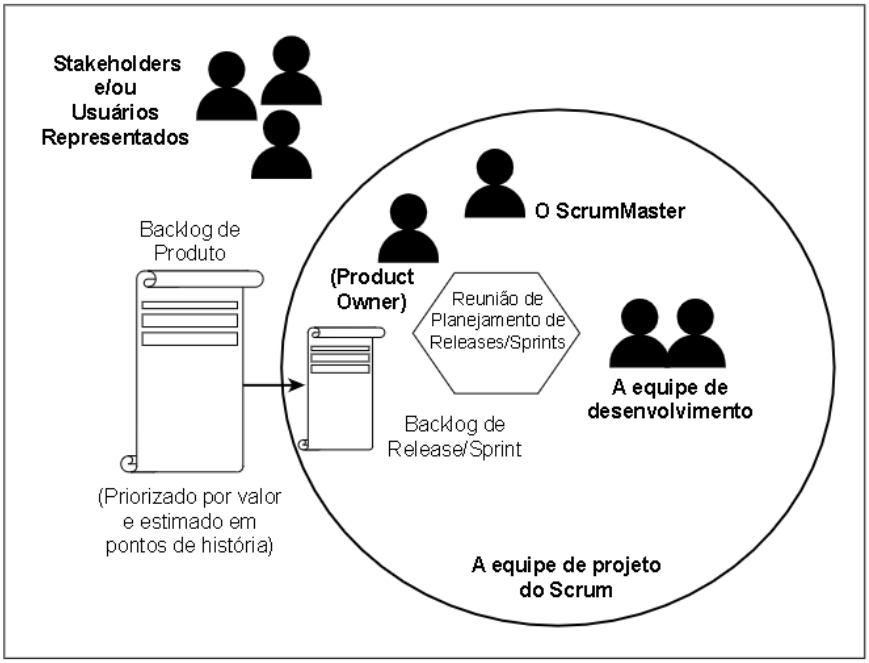
A metodologia Scrum tem seus princípios entrelaçados com o manifesto ágil, tendo como contemplação as atividades estruturais, sendo elas os requisitos, a análise, o projeto, a evolução deste projeto e a entrega final, ocorrendo estre processo dentro de um padrão denominado Sprint. A Sprint inicia-se como um ciclo de 2 a 4 semanas e tem uma certa quantidade de requisitos/histórias para serem introduzidas. (Pressman, 2011)

3.1 Funcionamento do Scrum

O funcionamento do Scrum segundo Pham (2012), deve ser composto por uma equipe versátil (ScrumMaster, um Product Owner (Dono do Produto) e equipe de desenvolvimento).

Tudo começa com o Product Owner, que é responsável por obter informações dos Stakeholders, ou usuários que os representem, para então elaborar uma lista de requisitos e criar um Backlog de Produto. Este processo pode ser observado na figura 2.

Figura 2 – Backlog de Produto, Backlog de Release e Backlog de Sprint.



Fonte: Scrum em Ação (Pham, 2012, pg8)

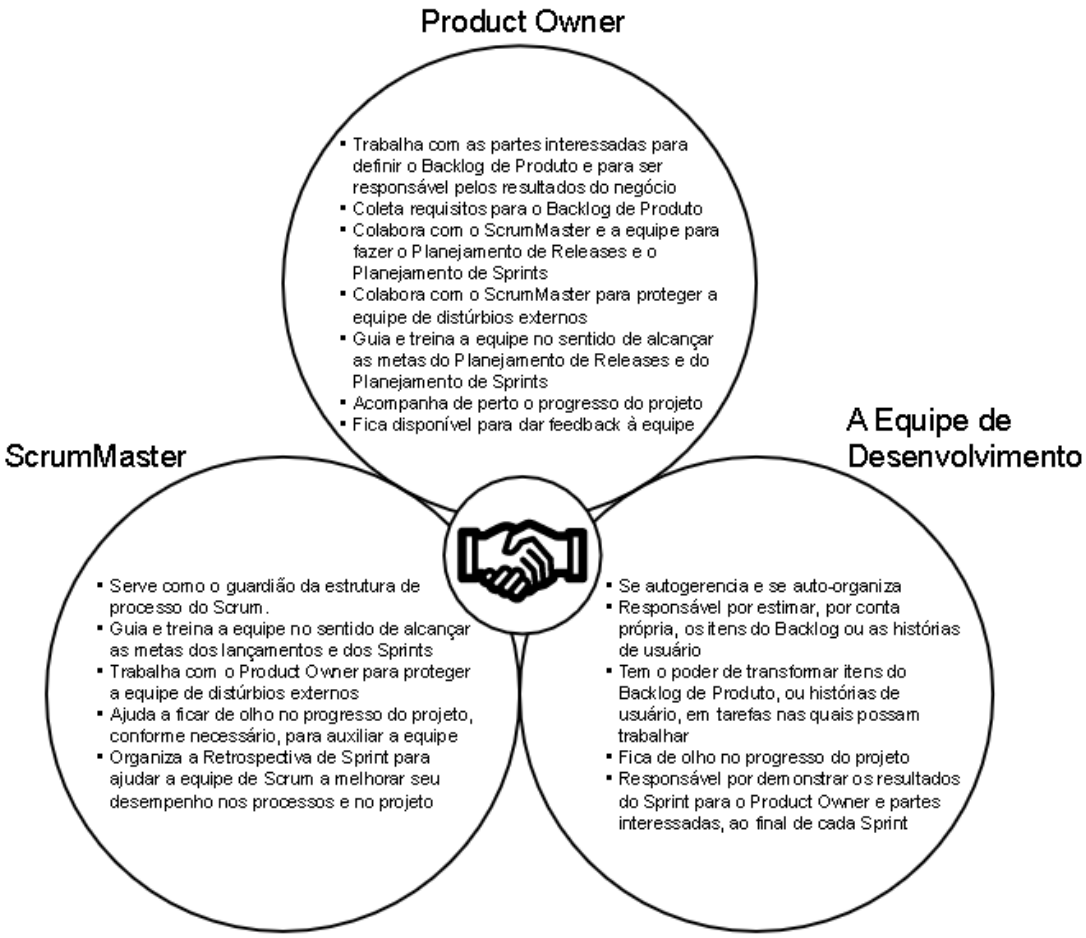
O Backlog de Produto é uma lista de requisitos priorizada, que pode incluir aspectos do negócio, tecnologias, questões técnicas e correções de bugs.

Antes do final de cada Sprint, a equipe se reunirá com o Product Owner para percorrer o que é conhecido como uma Revisão de Sprint, organizada pelo ScrumMaster. O Objetivo desta reunião é discutir o que foi e não feito, obter feedback e atualização do PO relativas a quaisquer mudanças na direção do mercado ou do produto.

Após a revisão do Sprint, mas antes do Sprint seguinte, a equipe de Scrum também se reunirá para percorrer uma Retrospectiva do Sprint, com o objetivo de identificar o que funcionou e o que não funcionou durante o Sprint atual.

Na figura 3 pode-se identificar um gráfico geral das responsabilidades colaborativas dos membros da equipe de Scrum para um melhor entendimento.

Figura 3 – Tudo se resume à colaboração entre a equipe



Fonte: Scrum em Ação (Pham, 2012, pg13)

4 QUALIDADE DE SOFTWARE

De forma bastante breve, atualmente podemos entender a preocupação com qualidade de software como fator crucial para que seja garantido que o produto final alcance as expectativas do cliente.

Segundo Bartié (2002, p25) “Qualidade não é uma fase do desenvolvimento de software, é parte de todas as fases. ”. Dentro do grande espectro que a qualidade do software envolve, podemos analisá-la a partir de dois pontos de vista principais: o ponto de vista do desenvolvedor e o ponto de vista do cliente. Para o desenvolvedor, um software de qualidade é aquele que faz uso de técnicas adequadas e boa prática de desenvolvimento com o intuito único de atender as exigências do cliente, já para o cliente, um software de qualidade é o que traz retorno e em geral, possui baixo custo.

No ramo da tecnologia, a qualidade de software pode ser resumida também como a redução do retrabalho, uma vez que, quando o projeto é desenvolvido corretamente desde o início da sua execução, as probabilidades de haver a necessidade de refazer o projeto são consideravelmente diminuídas.

Foi realizado a medição da qualidade por meio da verificação dos resultados, verificando erros antes da entrega e defeitos que acabam passando despercebidos e repassados para produção. Desta forma é possível garantir que o trabalho fosse realizado de forma correta.

4.1 Princípios influentes

Pressman (2011) também elucida os fatores de qualidade de McCall, Richards e Walters (1977) criadores uma categorização dos princípios que influenciam a qualidade de software:

- **Correção:** o quanto o software satisfaz e compre seus objetivos;
- **Confiabilidade:** o quanto pode-se confiar na realização das funções esperadas com a precisão exigida pelo software;
- **Eficiência:** a quantidade de código e recursos para garantir que o software execute sua função corretamente;
- **Integridade:** o quanto seguro os dados e acessos por pessoas não autorizadas pode ser controlado neste software;

- **Usabilidade:** o quanto de esforço se faz necessário para garantir a operabilidade e aprender a operar, preparar entradas e interpretar as saídas produzidas pelo software;
- **Facilidade de manutenção:** o quanto de esforço se faz necessário para identificar e eliminar problemas do software;
- **Flexibilidade:** o quanto de esforço se faz necessário para realizar modificações em um software em operação;
- **Testabilidade:** o quanto de esforço se faz necessário para realizar o teste deste software de modo a garantir o desempenho esperado;
- **Portabilidade:** o quanto de esforço se faz necessário para transferir o software de um ambiente para outro;
- **Reusabilidade:** o quanto um software pode ser reutilizado em outros programas;
- **Interoperabilidade:** o quanto de esforço se faz necessário para integrar um sistema a outro.

5 TESTES ÁGEIS

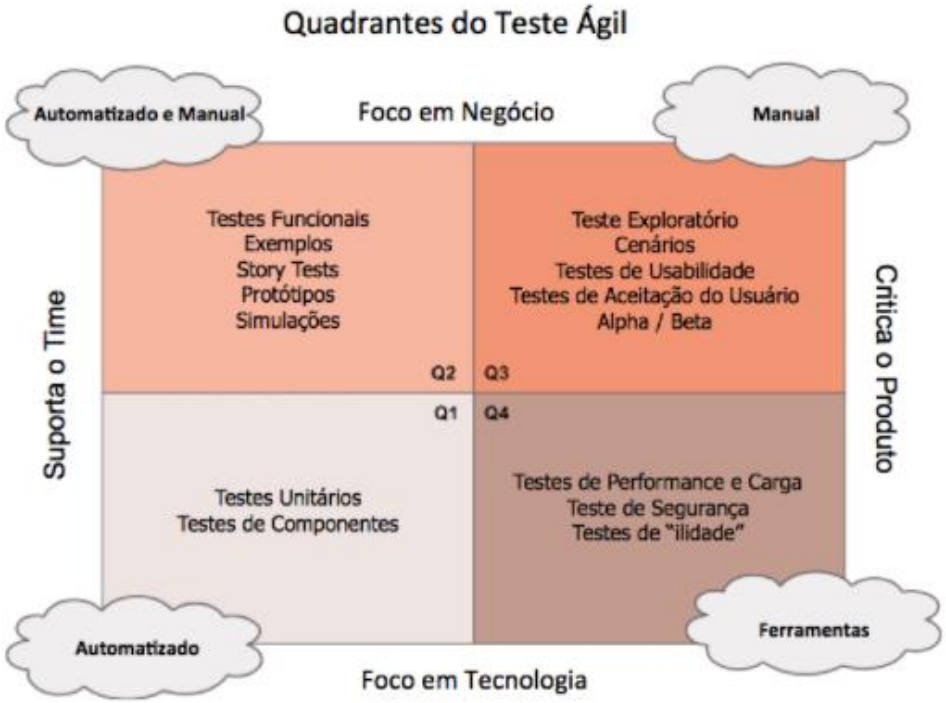
A partir do momento em que os releases/versões funcionais são liberados, existe a necessidade de testá-los. A cada nova release, um novo teste deve ser executado. O propósito geral do teste de software é verificar se o produto corresponde às funcionalidades esperadas e atende às necessidades do usuário final.

Os testes ágeis são, basicamente, testes de software condizentes com as regras do supracitado manifesto ágil e a sua utilização otimiza e torna precisa a detecção de defeitos no projeto, que poderiam não ser localizados manualmente antes de cada lançamento.

Crispin e Gregory (2009) enfatizam que o analista de teste tem uma grande transformação durante o decorrer do processo que faz com que ele trabalhe com mais proatividade e com maior comunicação e interação com os desenvolvedores e interessados do projeto.

O Quadrante de Teste Ágil foi criado por Brian Marick (Figura 4) e tem o intuito de orientar a realização dos testes e não necessariamente de servir como um padrão para tal.

Figura 4 – Quadrante do Teste Ágil



Fonte: Lisa Crispin. (2009)

Segundo Crispin e Gregory (2009) os quatro quadrantes podem ser explicados da seguinte forma:

- **Quadrante 1 –** realiza testes de unidade para validar uma menor parte do software e testes de componentes para a validação de uma parte maior da aplicação.
- **Quadrante 2 –** tem como objetivo realizar testes que o cliente possa entender, mas mantendo o foco no desenvolvimento. O cliente pode definir exemplos para serem utilizados como um melhor entendimento funcional da aplicação. Os testes são trabalhados de forma onde o cliente ou pessoas ligadas ao negócio possam entender. Os testes também podem ser automatizados, utilizando técnicas como a do nosso trabalho BDD.
- **Quadrante 3 –** é utilizado para criticar a aplicação como um usuário comum da vida real. É utilizado técnicas de conhecimento comum e intuição. O cliente pode transferir um feedback mais claro com a utilização deste quadrante. O teste exploratório é um ponto focal deste quadrante.
- **Quadrante 4 –** são testes mais técnicos, analisando performance, segurança e carga por exemplo. As técnicas são as mais variadas, desde níveis mais baixos a níveis mais altos.

6 BEHAVIOR DRIVEN DEVELOPMENT (BDD)

A conveniente e usual sigla BDD significa *Behavior Driven Development* ou, em português, Desenvolvimento Guiado por Comportamento é um resposta ao TDD (*Test Driven Developmente* ou, em português, Desenvolvimento Orientado por Testes), apresentado por Dan North em 2003.

Ainda segundo North (2003) o TDD é a técnica de desenvolvimento do software baseado em testes que são escritos antes do código, ou seja, se baseia em pequenos ciclos de repetições onde, para cada funcionalidade do sistema, um teste é criado antes, já que o BDD tem por objetivo testar um código a partir de um conjunto de cenários que descrevem como a aplicação deverá se comportar em determinada situação.

A técnica de BDD e suas práticas incluem envolver todas as partes do time no processo para um conhecimento único e claro do que é solicitado pelo cliente, e tem por objetivo produzir um software que atenda às expectativas do usuário final; fazer uso de linguagem do dia a dia para descrever o comportamento das aplicações; automatizar os exemplos com o intuito de obter rápido feedback; utilizar *should* para descrever o comportamento do software de forma que esclareça as responsabilidades e permita que as suas funcionalidades sejam questionadas e, por fim, usar dublês de teste (*stubs , mocks, dummies , fakes, spies*, etc) afim de colaborar com módulos e códigos que ainda não foram escritos.

O ápice do BDD é tornar a escrita e a compreensão de novos testes viáveis para todos os integrantes da equipe e inserir em todos os cenários, ou seja, em todas as interações do homem com o computador, passos lógicos e simples de como obter um resultado específico a partir de uma sequência de ações. Dessa forma os desenvolvedores passam a ter foco nas razões pelas quais os códigos devem ser criados e não em detalhes técnicos.

BDD se originou de práticas ágeis já estabelecidas tendo como foco tornar estas práticas mais acessíveis e efetivas para novas equipes de desenvolvimento ágil. Conforme destacada North (2003) o BDD foi evoluindo até chegar a um quadro mais amplo de análise ágil.

North (2003) estabelece que as especificações extraídas no fornecimento do cliente durante a coleta dos requisitos, devem ser utilizadas como linguagem utilizada no BDD, não engessando o *template* e o mantendo como artificial até o ponto de se criar restrições para os analistas, mas dar aos requisitos estrutura suficiente quebrando a história em fragmentos

constituintes para automatizá-las. Os critérios de aceite foram transformados em cenários assumindo a seguinte forma:

Given – Um contexto é iniciado.

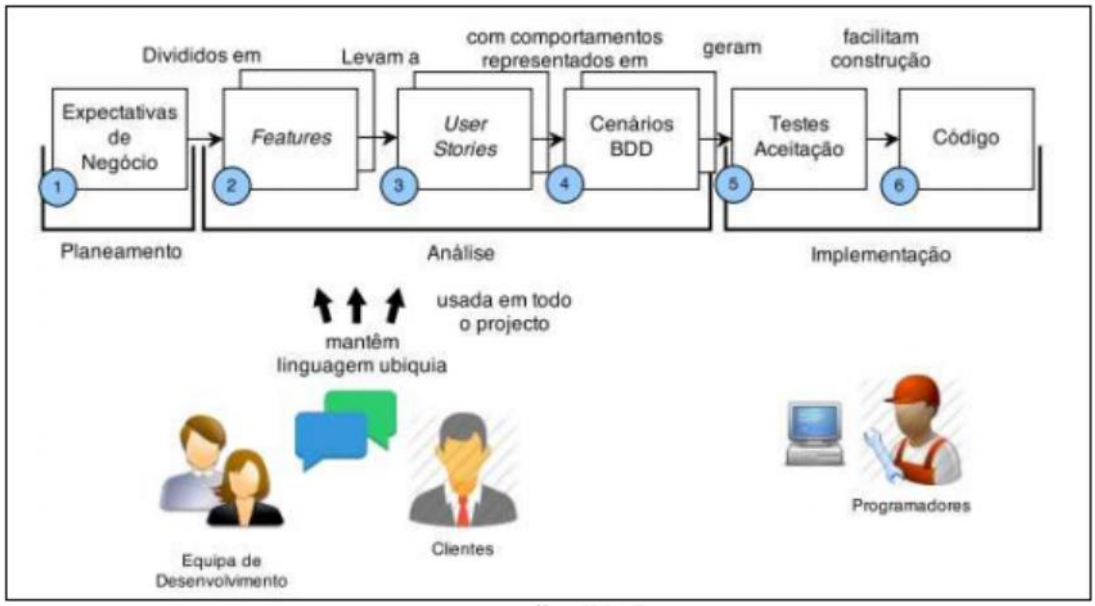
When – Quando algum evento ocorrer.

Then – Então analisar os resultados.

Smart (2015) afirma que BDD envolve comunicação e interação entre todas as partes envolvidas, mantendo uma visão clara e única dos requisitos solicitados disponibilizando um processo junto a ferramentas para promover uma boa comunicação entre o time e os demais interessados no projeto.

Silva (2014) ressalta os seis passos onde a técnica de BDD pode ser demonstrada, conforme pode-se analisar na figura 5. Este processo tem seguimento até o sexto passo, posteriormente o produto é validado pelos *stakeholders* e em seguida o ciclo é reiniciado novamente, alterando ou não o produto produzido anteriormente.

Figura 5 – Ilustração demonstrativa do processo do BDD.



Fonte: Silva. Uma Abordagem Ágil (2014)

7 ASPECTOS POSITIVOS E NEGATIVOS DA APLICAÇÃO DO BDD NO DIA A DIA

Buscando compreender e contextualizar a aplicação diária do BDD, o presente artigo adotou uma metodologia prática.

Foi realizado um estudo de caso na supracitada empresa multinacional "ABC", com sede brasileira na cidade de Uberlândia/MG. Um questionário constituído por 10 (dez) perguntas foi respondido por 13 colaboradores do Projeto A.

Os colaboradores que participaram deste questionário possuem um papel fundamentel, pois fornecem percepções e interpretações sobre o assunto, assim como fontes para busca de evidências corroborativas (YIN, 2001). Pode-se analisar as respostas a baixo no quadro 1.

Quadro 1 – Questionário enviado aos colaboradores do projeto A

Questão 01 – Qual a sua função dentro do Projeto A?	**Respostas**: 1 (um) Product Owner (PO); 1 (um) Scrum Master (SM); 2 (dois) Arquitetos de Software (Arq.); 4 (quatro) Analistas de Testes (QA) e 5 (cinco) Desenvolvedores (Dev)
Questão 02 – Qual a sua experiência com desenvolvimento de software ágil:	**Respostas**: PO – 11 anos; SM – 6 anos; Arq. 1 – 4 anos; Arq. 2 – 2 anos; QA 1- 1 ano; QA 2- 3 anos; QA 3 – 6 meses; QA 4 – 5 anos; Dev 1 – 7 anos; Dev 2 – 1 ano; Dev 3 – 3 anos; Dev 4- 9 meses; Dev 5 – 2 anos.
Questão 03 – Você conhece e compreende a técnica de BDD? Já usou essa ténica em outro(a) projeto/empresa?	**Respostas**: Todos os colaboradores afirmaram conhecer a técnica de BDD, no entanto, apenas o PO e o Dev 4 já haviam utilizado referida técnica em outras empresas

Questão 04 – Com a aplicação do BDD, o time apresentou melhor entendimento sobre os requesitos do cliente? Justifique. ***Continuação Questão 04***	**Respostas**: 11 colaboradores afirmaram obter melhor compreenção sobre o que lhes foi requisitado. Notaram que houve maior detalhamento dos requisitos, fato que auxiliou o início do desenvolvimento e dos cenários de testes. Destacaram, ainda, a necessidade de revisar algumas histórias de usuários, cuja aquelas que encontravam-se em formato BDD foram resolvidas com maior facilidade e agilidade. 2 colaboradores afirmaram não indentificar melhorias em relação à tecnica utilizada antes da introdução do BDD. De forma geral, foi possível observar que o PO, os Arquitetos e o Scrum Master tiveram melhor entendimento sobre o que havia sido requisitado. Os desenvolvedores e analistas não notaram significativo impacto.
Questão 05 – Você pôde perceber melhora na comunicação no decorrer do projeto com a introdução do BDD? Justifique.	**Respostas**: 10 colaboradores afirmam acreditar que os membros da equipe tiveram comunicação mais efetiva, por alcançarem a mesma compreensão sobre o requisito. Destacaram também que o formato BDD estimulou a discussão em relação aos detalhes, como os critério de aceitação/cenários entre o time. 3 colaboradores não identifica o BDD como influenciador na unificação do time ou não reconheciam problemas de intercomunicação.

Questão 06 - Após a introdução do BDD, qual foi a sua perspectiva em relação à retrabalho? Justifique. ***Continuação Questão 06***	**Respostas**: 11 colaboradores identificaram melhorias em relação ao retrabalho. Afirmam que os cenários foram desenvolvidos de forma mais detalhada, fato que beneficiou a execução dos testes de software, uma vez que houve significativa queda na quantidade de retestes, defeitos encontrados e tempo hábil utilizado para a explicação dos problemas encontrados para terceiros. Apenas 2 colaboradores afirmaram não ter parâmetros comparativos para definir se, de fato, a técnica de BDD, por si só, garante o fim do retrabalho
Questão 07 - Numa escala de 0 a 5, sendo 0 nenhuma mudança e 5 mudanças além das expectativas, qual a sua nota na eficácia do desenvolvimento do projeto após a introdução do BDD? Justifique**.**	**Respostas**: 9 colaboradores atribuiram nota 5 e afirmaram identificar comunicação efetiva, redução dráticas de retrabalho, maior detalhamento e melhor entendimento comum dos requisitos; 2 colaboradores atribuiram nota 4 e afirmaram ter notado aspectos positivos apenas no que diz respeito à diminuição de retrabalho e clareza nos requisitos; 2 colaborares atribuiram nota 2 e afirmaram não identificar vantagens significativas, além da melhor compreensão entre o time.
Questão 08 - Quais foram as vantagens e desvantagens do uso da técnica de BDD no projeto? Justifique.	**Respostas**: Todos os colaborares afirmam identificar alguma vantagem na utilização da técnica de BDD. Os pontos mais destacados entre os participantes foram: ganho na descrição dos cenários de negócio no início do desenvolvimento, melhor comunicação entre o time, entendimento comum do requisito e menos retrabalho. Apenas 4 colaboradores afirmar não ter parâmetros de comparação em relação à experiências anteriores com comunicação e retrabalho pelo curto período na área tecnológica.

Questão 09 - Depois da introdução do BDD, você notou se houve melhora na intercomunicação entre os profissionais que compõe a equipe? Justifique.	**Respostas**: 9 colaboradors afirmam identificar melhora na comunicação entre as equipes e ganho em relação à sincronização das informações, além do estímulo à discussão em pontos falhos; 4 colaboradores afirmar não terem notado diferença na intercomunicação da equipe, justamente por não identificar falhas anteriores neste quesito; Todos os QA notaram melhoria na comunicação da equipe.
Questão 10 - Na sua concepção, a técnica de BDD é útil para o desenvolvimento ágil do projeto ou o seu uso pode ser considerado irrelevante? Justifique.	**Respostas**: 11 colaboradores identificam positivamente o uso do BDD como técnico de desenvolvimento ágil de software. Afirmam que esta auxilia na compreensão dos cenários, das necessidades de negócio e melhor qualidade do produto final; 2 colaboradores afirmar que o uso do BDD não agrega valor significativo ao projeto.

Fonte: Elaborado pelo próprio autor

CONSIDERAÇÕES FINAIS

Diante dos conceitos teóricos e práticos neste estudo abordados, é possível concluir que a utilização do BDD em projeto de desenvolvimento ágil, de forma geral, agrega grande valor durante todo o processo.

A preocupação com qualidade de software faz parte do cotidiano do projeto A, que após a implantação da técnica de BDD observou aspectos bastante positivos, principalmente no que diz respeito à intercomunicação da equipe, redução do retrabalho e, consequentemente, maior eficácia e agilidade na entrega do produto final ao cliente.

Todos os profissionais que participaram do estudo neste trabalho desenvolvido identificaram vantagens na utilização do BDD e um dos pontos por eles destacados foi a riqueza

de detalhes que puderam ser observados na execução dos cenários que, além de proporcionar melhor e mais ágil entendimento do que lhes são propostos, garantiu o aumento da qualidade no desenvolvimento do projeto como um todo.

Contudo, a utilização efetiva da técnica de BDD pode ser vista como uma importante ferramenta de testagem que escreve códigos, cujo objetivo principal é melhorar as funcionalidades, que vão ao encontro das necessidades do cliente final,

Podemos entender o BDD também como a aproximação da empresa contratada com o consumidor final do produto, neste caso, do projeto que oferece, sendo uma técnica facilitadora da comunicação dos profissionais que integram o corpo desenvolvedor de referido projeto e, desta forma, garante que a necessidade do consumidor/usuário final seja suprida.

Apesar de não ser o intuito principal do presente trabalho, este pode ser utilizado como base argumentativa para que a técnica de BDD possa ser aplicada em diversos projetos desta e de outras empresas de tecnologia que priorizem agilidade e qualidade de software.

REFERÊNCIAS BIBLIOGRÁFICAS

AGILE MANIFESTO. **Manifesto for Agile Software Development.** <http://agilemanifesto.org/>. Acesso em: fevereiro de 2021.

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: Fevereiro de 2021.

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.

KOSCIANSKI, A., SOARES, M. (2006) **Qualidade de Software. 2ª edição.** São Paulo: Novatec.

NORTH, Dan. **Introducing BDD**. 2003. Disponível em:<http://dannorth.net/introducing-bdd/> Acesso em: Março de 2021.

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: Março de 2021.

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.

SILVA, António. **Uma Abordagem Ágil.** Universidade Nova de Lisboa, Portugal. 2014.

SMART, John F. **BDD in Action.** Manning Publications Co. NY. 2015.

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

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