UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

KARLA MICHELE BARBOSA DA SILVA

TEST DRIVEN DEVELOPMENT E EXTENSÕES: UMA ANÁLISE DO MERCADO BRASILEIRO

TRABALHO DE GRADUAÇÃO

Recife

2016

KARLA MICHELE BARBOSA DA SILVA

TEST DRIVEN DEVELOPMENT E EXTENSÕES: UMA ANÁLISE DO MERCADO BRASILEIRO

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientadora: Carla Taciana Lima Lourenço Silva Schuenemann

Recife

2016

KARLA MICHELE BARBOSA DA SILVA

TEST DRIVEN DEVELOPMENT E EXTENSÕES: UMA ANÁLISE DO MERCADO BRASILEIRO

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Aprovado em _____ de _____ de _____.

BANCA EXAMINADORA:

Carla Taciana Lima Lourenço Silva Schuenemann

_____

Simone Santos

_____

Dedico este trabalho primeiramente a Deus, pois sem Ele, nada seria possível. E aos meus pais, que sempre me encorajaram na luta pelos meus sonhos.

AGRADECIMENTOS

Primeiramente agradeço a Deus pela oportunidade de estudar em uma excelente universidade pública, pela sabedoria e força para superar cada fase dessa minha trajetória acadêmica.

Em segundo lugar, gostaria de expressar a minha gratidão, aos meus pais, Neide Cordeiro e Carlos Roberto. Agradeço a eles por nunca medirem esforços em apoiar os meus sonhos, por serem meus exemplos de caráter, humildade e bondade e por sempre terem aquela palavra de apoio que muitas vezes me motivou a continuar nessa jornada, mesmo diante das dificuldades. Agradeço também a minha irmã, Thays Silva, que sempre esteve presente, sendo uma amiga e me ajudando com os projetos mesmo sem entender muita coisa de computação.

Não poderia deixar de agradecer ao Centro de Informática e a Universidade Federal de Pernambuco por tantas oportunidades concedidas durante a minha trajetória da graduação. Agradeço também a todos os professores do centro, que contribuíram de forma significante com o meu crescimento profissional e pessoal. Em especial, agradeço a minha orientadora, professora, Carla Silva, por todo apoio e críticas durante esta última etapa da graduação.

Agradeço também aos meus colegas de turma, por toda ajuda e companheirismo nessa trajetória de um pouco mais de quatro anos. Em especial, agradeço a Isabel Amaral, Fagner Fernandes, Karine Gomes, Karina Moura e Marcela Oliveira por todo apoio. Cada projeto, cada disciplina, um grupo diferente com novos desafios, que me ensinaram muito sobre trabalhar em grupo.

Igualmente, não poderia deixar de agradecer ao CESAR, que me deu a oportunidade de colocar em prática toda a teoria aprendida na academia. Agradeço por toda a compreensão da minha equipe e colegas do trabalho durante os momentos de aperto da graduação.

Enfim, agradeço a todos que participaram dessa etapa da minha vida acadêmica com palavras de motivação, apoio nos projetos e força para não desistir da caminhada. Sou muito grata por tudo!

"Computers are magnificent tools for the realization of our dreams, but no machine can replace the human spark of spirit, compassion, love, and understanding."

Louis Gerstner

RESUMO

O processo de desenvolvimento de software é um processo complexo que envolve muitas etapas, entre elas: elicitação de requisitos, codificação e testes. As atividades realizadas em cada etapa buscam o mesmo objetivo, entregar um software de qualidade. Para isso, a indústria e a academia tem se esforçado bastante em busca de novas práticas que auxiliem e contribuam para que esse objetivo seja alcançado. Hoje em dia, uma prática muito utilizada nas empresas que adotam metodologias ágeis é o *Test Driven Development*. No entanto, existem muitas outras práticas estão se popularizando, entre elas: *Acceptance Test-Driven Development* e *Behavior-Driven Development*. Cada prática listada acima tem uma abordagem diferenciada e suas especificidades contribuem para um processo de qualidade. Este trabalho tem o objetivo de entender melhor quais os benefícios, mudanças identificadas após a adoção e dificuldades encontradas no processo de adoção do TDD e extensões em projetos de software de empresas brasileiras. Para obter esses dados, foi realizada uma pesquisa com 69 profissionais da indústria de software. Através da pesquisa, foi possível identificar que o TDD é a prática mais usada pela indústria brasileira e o ATDD a menos usada. Além disso, os benefícios, dificuldades e mudanças mais comentados pelos entrevistados foram comparados com os já existentes na literatura, e verificou que alguns deles ainda não tinham sido citados pela academia ou indústria.

Palavras-chave: desenvolvimento de software, qualidade de software, metodologias ágeis, *Acceptance Test-Driven Development*, ATDD, *Behavior-Driven Development*, BDD, *Test Driven Development,* TDD

ABSTRACT

The software development process is a complex process that is composed by many phases, for instance: requirements elicitation, coding and testing. The activities from each level try to achieve the same goal, deliver software with high quality. For that purpose, industry and universities are working hard to find new practices to reach that goal. Nowadays, a practice widely used in companies that adopt agile methodologies is the Test Driven Development; however, other practices are becoming popular too, for example: Acceptance Test-Driven Development and Behavior-Driven Development. Each practice listed above has a different approach and its specific features contribute to a quality process. This research aims to understand what benefits, changes identified after the adoption and difficulties encountered in the process of adoption of the TDD and extensions in software projects of Brazilian companies. To collect that information, a survey was conducted with 69 professionals of the software industry. With that survey, it was possible to recognize that TDD is the practice mostly used by Brazilian companies, but ATDD is rarely used. In addition, the main benefits, challenges and changes commented by the respondents were compared with the ones from the existing literature, and we noticed that some of them had not yet been cited by academia or industry.

Keywords: software development, software quality, agile methodology, Acceptance Test-Driven Development, ATDD, Behavior-Driven Development, BDD, Test Driven Development, *TDD*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATDD	Acceptance Test-Driven Development
BDD	Behavior-Driven Development
DbC	Design by Contract
PO	Product Owner
TDD	Test-Driven Development
TI	Tecnologia da Informação
SQA	Software Quality Assurance
STDD	Story-Test-Driven Development
XP	eXtreme Programming

SUMÁRIO

1 INTRODUÇÃO

A engenharia de *software* tem como objetivo desenvolver software de qualidade dentro dos custos estabelecidos (SOMMERVILLE, 2007). No entanto, existem muitas variáveis e atividades associadas ao desenvolvimento de um *software*. Por isso, academias e organizações estão sempre buscando práticas e processos que melhorem o processo de desenvolvimento.

Putnam-majarian e Putnam (2015) comentam que melhores práticas sempre são recomendadas quando um projeto de *software* é iniciado. Apesar de diminuir o número de projetos que falham, essas melhores práticas não garantem o sucesso do projeto. Uma vez que, alguns projetos continuam falhando. De acordo com o Project Management Institute (2013, p.35), o sucesso de um projeto “deve ser medido em termos da sua conclusão dentro das restrições de escopo, tempo, custo, qualidade, recursos e risco, conforme aprovado entre os gerentes de projetos e a equipe sênior de gerenciamento”. Sempre tendo em vista, as necessidades do cliente.

Almejando mitigar alguns dos riscos de uma possível falha do projeto, as organizações investem em alternativas e novas soluções para garantir um produto de qualidade. Essas atividades são voltadas para que a confiabilidade do software seja atingida e mostrar que o software está em condições de uso (SOMMERVILLE, 2007). Essas alternativas comtemplam novas ferramentas, novos processos de testes e novas técnicas associadas a fase de validação de *software*. No entanto, Dijkstra e outros (1972) afirmam que os testes não garantem a ausência de erros, mas podem mostrar a sua presença.

Como resultado da busca pela qualidade, Beck (2002) apresenta o *Test-Driven Development (TDD),* que segundo ele é uma forma de guiar o desenvolvimento com testes automatizados. Ele classifica o TDD como um novo estilo de desenvolvimento. Esse novo estilo de desenvolvimento é baseado no TDD mantra, *red/green/refactor.* Onde o objetivo final é escrever um código enxuto que funcione.

Com esse novo estilo de desenvolvimento, o TDD tornou-se rapidamente popular na academia e indústria. Sob o mesmo ponto de vista, Ambler (2013) afirma em um de seus artigos, que o TDD está sendo rapidamente adotado por desenvolvedores que utilizam metodologias ágeis na indústria de software. Essas empresas que fazem uso de metodologias ágeis, seguem o Manifesto Ágil proposto por Beck e outros (2001), que defendem "*Working software over comprehensive documentation*" e "*Customer collaboration over contract negotiation*". Esse manifesto e seus princípios se adequam perfeitamente ao estilo de desenvolvimento pregado pelo TDD.

No entanto, com o propósito de buscar cada vez mais pela qualidade, outras técnicas foram surgindo a partir do TDD. Oram e Wilson (2011), associam o (TDD) a uma pílula que contém algumas variantes oficiais. São elas: *Acceptance-Test-Driven Development* (ATDD) e *Behaviour-Driven Development* (BDD), cada prática citada acima, as também conhecidas como extensões do TDD, possuem suas especificidades.

1.1 MOTIVAÇÃO

Com tantas novidades no ambiente de desenvolvimento de software, em 2012, foi publicado o artigo "*Test Driven Development: The State of the Practice*" (Hammond e Umphress, 2012) com o objetivo de entender a aceitação e efetividade do TDD e suas extensões. Como resultado, os autores identificaram que trabalhos futuros seriam necessários para entender se estavam desenvolvendo as funcionalidades corretas e para uma melhor fundamentação sobre as descrições e definições das técnicas envolvidas na pesquisa.

Diferente do artigo de Hammond e Umphress (2012) que focou nos conceitos, aceitação e limitações, este trabalho será desenvolvido com o apoio de uma pesquisa online respondida por empresas de desenvolvimento de software atuantes no Brasil. Nesse contexto, essa pesquisa visa identificar as principais mudanças causadas pela adoção das técnicas, as dificuldades encontradas, os benefícios, e os fatores que foram críticos para o sucesso do projeto. Um formulário com treze perguntas ajudará a identificar esses três pontos críticos e, possivelmente, fazer um levantamento sobre o perfil dessas empresas e projetos.

1.2 OBJETIVOS

Diante de tantas práticas que buscam a qualidade em produtos de software, esse trabalho tem como objetivo geral identificar e analisar o uso no TDD e extensões no contexto brasileiro. Para atingir tal objetivo geral, os seguintes objetivos específicos foram definidos:

- Identificar as mudanças causadas pelo uso do TDD e suas extensões, sejam elas de processo, métricas ou produto;
- Identificar as principais dificuldades encontradas na adoção das técnicas;
- Identificar os benefícios percebidos pelos colaboradores após a adoção das técnicas;

Além disso, será possível identificar o perfil das empresas brasileiras que fazem uso das práticas, e possíveis ferramentas usadas pelos times.

1.3 ESTRUTURA DO DOCUMENTO

Contando com o capítulo de introdução, que apresenta a motivação e os objetivos da pesquisa, o presente trabalho está organizado em cinco capítulos. São eles:

Capítulo 2: apresenta todo o referencial teórico fundamental para o entendimento desse trabalho. Esses conceitos estão relacionados ao processo de desenvolvimento de software, qualidade de software e práticas usadas na engenharia de software, entre elas, ATDD, BDD e TDD.

Capítulo 3: descreve detalhadamente a metodologia de pesquisa usada no trabalho, a classificação da pesquisa, os critérios usados na seleção dos entrevistados e a forma como eles foram selecionados.

Capítulo 4: demonstra os resultados colhidos da pesquisa. Esse resultados foram analisados e interpretados para que possam responder as perguntas de pesquisa do presente trabalho.

Capítulo 5: apresenta as considerações finais sobre o projeto, quais foram as limitações e quais as recomendações para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

O presente capítulo foi dividido em seis tópicos, que representam os conceitos fundamentais para o entendimento dessa pesquisa. Os três primeiros sub-tópicos são assuntos mais gerais da engenharia de software, como por exemplo, processos de software, metodologias ágeis e qualidade de software. No entanto, os demais apresentam uma introdução as práticas estudas nesse trabalho. São elas: TDD, ATDD e BDD.

2.1 PROCESSOS DE SOFTWARE

Segundo Sommerville (2007, p. 6): “Um processo de software é um conjunto de atividades e resultados associados que produz um produto de software”. Esse processo é considerado um processo complexo, por contemplar várias atividades que muitas vezes estão sobrepostas.

Cada projeto com suas especificações pode adotar um processo de desenvolvimento diferente. No entanto, algumas atividades são comuns a todos os processos de desenvolvimento de software. São elas: especificação dos requisitos, projeto e implementação, validação e evolução do sistema (SOMMERVILLE, 2007). Os diversos modelos de processo de software tentam oferecer uma representação abstrata e genérica do processo.

O modelo cascata, por exemplo, sugere uma abordagem sequencial e sistemática para o desenvolvimento. Esse é bem tradicional e está presente na engenharia de software há muito tempo (PRESSMAN, 2011). Uma das maiores críticas com relação a esse modelo é a falta de flexibilidade. Para Sommerville (2007, p. 45), “os compromissos devem ser assumidos no estágio inicial do processo, o que torna difícil reagir as mudanças de requisitos do usuário”.

Já o modelo incremental, não oferece um processo totalmente linear semelhante ao modelo cascata. O modelo incremental defende que o fornecimento de versões funcionais e sequenciais para o usuário é importante. Sendo assim, o modelo propõe que vários incrementos sejam entregues e que feedbacks de

iterações anteriores podem ser usados na iteração atual (LASKOSKI; RADAELLI, [2012]).

Por último, o desenvolvimento evolucionário baseia-se que o software vai evoluir ao longo do tempo e que mudanças nos requisitos podem e vão acontecer. O modelo apresenta uma proposta iterativa, onde o resultado vai sendo refinado pelos usuários. Para Sommerville (2007), muitas vezes o desenvolvimento evolucionário é mais eficaz do que o modelo em cascata.

Todos os três modelos são amplamente usados na indústria. Além disso, o projeto não precisa escolher apenas um modelo, mas sim usar combinações com as melhores práticas deles.

2.2 METODOLOGIAS ÁGEIS

Devido ao ambiente globalizado e próprio para mudanças que os negócios vivem hoje, empresas precisam responder a essas mudanças rapidamente. Logo, isso vai influenciar o software usado pela companhia, que geralmente é parte do negócio, e por isso deve acompanhar essas mudanças de forma rápida (SOMMERVILLE, 2007). Pensando nesse desenvolvimento rápido, na década de 1990, alguns desenvolvedores começaram a propor novos métodos para desenvolvimento de software, as chamadas metodologias ágeis.

Metodologias ágeis são usadas para otimizar o processo de desenvolvimento. O objetivo delas é que software de qualidade seja produzido em curtos intervalos de tempo (LIVERMORE, 2007). As metodologias ágeis surgiram como uma alternativa ao gerenciamento de projetos na forma tradicional. Onde boa parte do tempo era gasto na fase de planejamento de como o sistema deveria ser desenvolvido e a cada mudança, o retrabalho era muito custoso.

Existem algumas metodologias e *frameworks* que são globalmente conhecidos como ágeis. Para algumas delas, os requisitos podem e vão mudar, criando assim oportunidades de melhoria. Por isso, eles não são tratados como algo fixo. Pelo contrário, as funcionalidades são tratadas como estória de usuários, que é uma forma mais simples de descrever o objetivo de uma funcionalidade (WELLS,

2013). As metodologias e frameworks mais populares são: o *Scrum*, XP, *Crystal* e outras.

Segundo Schwaber e Sutherland (2013), o *Scrum* é um *framework* que visa ajudar pessoas a resolverem problemas complexos. Ao mesmo tempo que é produtivo e criativo, busca entregar produtos que agregam valor ao cliente.

Basicamente, o *Scrum* prega que o time deve ser composto pelo *Product Owner* (PO), o time de desenvolvimento, e o *Scrum Master* (pessoa responsável pelo entendimento do *framework* pelo time). Além disso, os times são auto-organizados e multifuncionais. Além dos papéis fundamentais de um time, o *Scrum* defende alguns eventos dentro de uma *Sprint*. Em primeiro lugar, uma *Sprint é* período de tempo pré-definido para desenvolvimento incremental de um produto usável, que é composta por alguns eventos menores. São eles: planejamento da Sprint, as reuniões diárias do *Scrum*, as revisões de Sprint e a retrospectiva. Por último, os artefatos gerados representam valor e garantem a transparência e oportunidades para adaptação (SCHWABER; SUTHERLAND, 2013).

A metodologia *Extreme Programming* foi desenvolvida durante um projeto na empresa Chrysler, enquanto recebia consultoria de Kent Beck. A partir daí, Beck trabalhou no seu refinamento até torna-la mundialmente conhecida (LIVERMORE, 2007). O XP enfatiza o trabalho em equipe, onde os times são auto gerenciáveis e buscam resolver os problemas de forma eficiente. Além disso, o cliente faz parte da equipe e está envolvido nas atividades de especificação e priorização de requisitos. Para Lindstrom e Jeffries (2004), o XP melhora os projetos através de quatro valores: comunicação, simplicidade, feedback e coragem.

Cockburn (2004) não acreditava que uma metodologia poderia ser usada em todos os projetos. Sendo assim, ele idealizou o Crystal em 1998. No entanto, o Crystal é na verdade uma família de metodologias composta por *Crystal Clear*, *Crystal Yellow*, *Crystal Orange* e outros (MCLAUGHLIN, [2015]). Segundo Livermore (2007), com *Crystal* cada organização usa apenas o tanto de metodologia que seus negócios precisam.

No inicio do século XXI, Beck e outros (2001) proclamaram o manifesto ágil, que defende quatro importantes valores: “*Individuals and interactions over processes*

and tools", "*Working software over comprehensive documentation*", "*Customer collaboration over contract negotiation*" e "*Responding to change over following a plan*". Além disso, existem 12 princípios seguidos pelo manifesto ágil, que guiam os times na implementação e execução de projetos ágeis.

2.3 QUALIDADE DE SOFTWARE

Com sistemas presentes em cada vez mais lugares e integrados em muitas das nossas atividades diárias, a busca pela qualidade aumentou para que as chances de achar uma falha em um sistema em produção diminuíssem. Segundo a International Standards Organization (2008 citados por PROJECT MANAGEMENT INSTITUTE, 2013), a qualidade é "o grau em que um conjunto de características inerentes atende aos requisitos". Tanto para a indústria de *software* como para outros ramos, a qualidade é um dos pontos críticos para o sucesso do produto. No entanto, o conceito de qualidade é muito subjetivo. De acordo com Rios e Moreira (2013, p.2) existem algumas características necessárias para um software ser considerado de qualidade. São elas:

> O número e a severidade de defeitos é aceitável pela organização; O software é entregue dentro do prazo/custos, atende os requisitos e/ou expectativas; Foi construído de tal maneira que possa ser mantido de forma eficiente apos sua implantação.

Glass (1998) ainda vai mais além, e diz que tudo isso é importante, mas se o usuário não estiver satisfeito, nada mais importa. No entanto, medir a qualidade não é uma tarefa fácil. Por isso, McCall, Richards e Walters (1977), escolheram onze fatores como fundamentais de qualidade. São eles: correção, confiabilidade, eficiência, integridade, usabilidade, facilidade de manutenção, flexibilidade, testabilidade, portabilidade, reusabilidade e interoperabilidade. Esses fatores são focados nas características operacionais, adaptação a novos ambientes e suporte a mudanças.

Ainda assim, a qualidade de software demanda muito esforço para ser alcançada. No entanto, alguns processos do gerenciamento da qualidade tem como objetivo garantir que as especificações da solução foram atendidas. Em especial, a garantia e o controle da qualidade. A garantia da qualidade ou *Software Quality*

Assurance (SQA) é um processo gerencial e deve ser aplicada durante todo o ciclo de desenvolvimento do produto. Tem como atividades desde a supervisão das atividades do controle da qualidade até definição de procedimentos. Já o controle da qualidade, está relacionado ao monitoramento dos resultados e aos feedbacks que a equipe pode precisar para ajustar o processo quando o mesmo não atende as metas estabelecidas para a qualidade (PRESSMAN, 2011).

Para certificar-se que o sistema atende ao que foi planejado, atividades de validação e verificação ocorrem durante todo o processo de software. Para Sommerville (2007), a validação é mais geral e está relacionada a garantia de que o sistema atende as necessidades do cliente. Já a verificação, vai verificar se as especificações (requisitos funcionais e não funcionais) estão sendo atendidos.

A execução de testes em um sistema é a principal abordagens do processo de validação e verificação. Além disso, é uma forma bastante popular para verificar se a aplicação atende às especificações (SOMMERVILLE, 2007). Nese caso, é importante lembrar que o nível de qualidade dos testes vai influenciar na medição da qualidade.

2.4 TDD

O *test-driven development* é uma técnica de programação popularizada por Kent Beck. Essa técnica geralmente é associada com métodos ágeis, em particular com o eXtreme Programming (AMBLER, 2013). Além disso, o TDD apresenta ciclos de desenvolvimento pequenos e os casos de testes são escritos antes mesmo da codificação da funcionalidade.

Semelhantemente, Koskela (2008, p.4) afirma que, "*Only ever write code to fix a failing test*". O autor ainda comenta que essa sentença resume basicamente o que é TDD. O *test-driven development* prega que os testes devem ser escritos primeiro. Então, só depois dos testes escritos, é que os desenvolvedores devem começar a escrever o código que fará os testes passarem. Isso é possível com a ajuda de *mocks* ou *stubs,* que são objetos que serão utilizados para simular o comportamento de objetos reais da aplicação.

Para Beck (2002), um dos grandes benefícios do TDD é que os próprios desenvolvedores aprendem a escrever os próprios testes unitários. Com isso, Koskela (2008) afirma que essa simples técnica ajuda a evitar muitos vícios de codificação.

De forma mais detalhada Beck (2002) explica que o ciclo do TDD se resume em três etapas, também conhecido como o TDD *mantra*, que visa um código enxuto que funcione. Para ele, a primeira etapa é relacionada a escrita do teste da mesma forma que os desenvolvedores imaginaram. Com as *interfaces* que eles supõem que teriam e todos os elementos contidos na estória. Feito isto, a segunda etapa é fazer o teste rodar. A barra de progresso do *framework* de teste deve ficar verde (Figura 1) e anotações podem ser feitas já pensando nas melhorias. Por último, a preocupação deve ser no melhoramento do código. Por exemplo, removendo código duplicado que podem ter sido introduzidos na fase anterior.

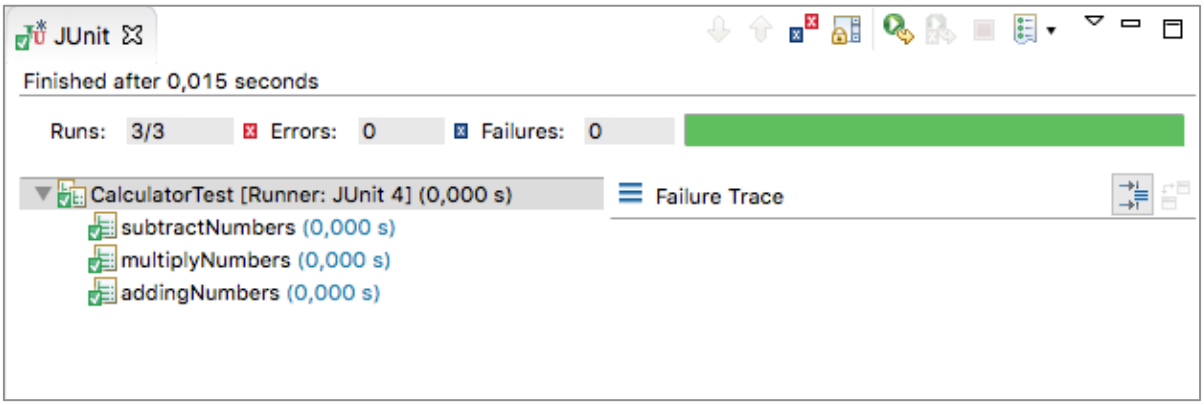

Figura 1: Exemplo de testes unitários executados com o JUnit

A figura 1 mostra exemplos de testes unitários na fase dois do ciclo de TDD. Onde é possível observar a barra verde na parte superior direita indicando que os testes passaram. Segundo Beck (2002), um bom teste unitário deve: ser rápido, rodar isolado, usar dados que o tornem fácil de ler e entender, usar dados reais, e representam um passo a frente no processo.

Para Serta (2011), a aplicação do TDD, geralmente se usa dois tipos de ferramentas: um *framework* de teste unitário e uma ferramenta para de criação de *mocks*. Algumas ferramentas comumente usadas são: JUnit [http://junit.org/],

TestNG [http://testng.org/doc/index.html], Mockito [http://mockito.org/], EasyMock [http://easymock.org/] e outras.

A partir do TDD, outras extensões foram surgindo em busca por melhores práticas. Entre elas: o ATDD, o BDD, e outras.

2.5 ATDD

Similar ao TDD, o *acceptance test driven development,* ou desenvolvimento orientado a testes de aceitação, também defende que os testes devem ser escritos primeiro. Inesperadamente, de acordo Agile Alliance (2016), a primeira menção do ATDD foi feita por Kent Beck em seu livro Test-Driven Development by Example, no entanto Beck a chamou de *Application Test-Driven Development e* não deu muita importância.

O ATDD é uma prática onde os requisitos são obtidos de forma colaborativa. Antes de codificar cada funcionalidade, membros do time colaboram criando exemplos. A partir daí, os times traduzem tudo em testes de critérios de aceitação.

Segundo Gärtner (2012), o maior problema no desenvolvimento de software é a comunicação. De acordo com o autor, as pessoas não comunicam claramente o que deseja, e o ATDD é um prática que vai ajudar a trabalhar esse desafio. A situação ideal proposta pela Agile Alliance (2016) é que *product owner,* cliente ou pessoas experts no domínio possam escrever testes de aceitação sem consultar os desenvolvedores. Sauvé e Abath Neto (2008) ainda descrevem outros benefícios do uso do ATDD. Por exemplo, sincronização entre requerimentos e código escrito e foco no interesse do cliente.

Hendrickson ([2008]), apresenta um ciclo de desenvolvimento para testes de aceitação composto por 4 etapas. São elas: debater, refinar, desenvolver e revisar (Figura 1).

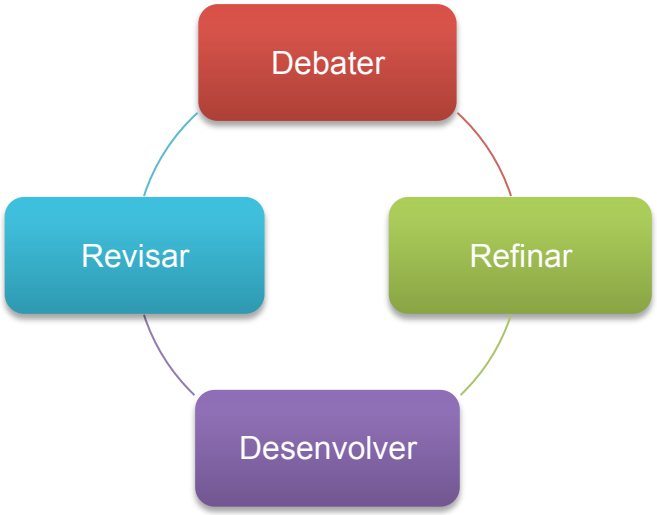

Figura 2: Ciclo do ATDD

Na etapa de debater, os participantes da reunião discutem as histórias dos usuários até entenderem o que o PO gostaria de ter como funcionalidade. Já na etapa de refinamento, o foco consiste em capturar os testes em formato que funcione com o framework de automação escolhido pelo time. A terceira etapa, consiste no desenvolvimento da funcionalidade para que os testes de aplicação possam passar. No entanto, se a equipe usar TDD, o desenvolvimento começa pelos testes unitários. Por último, na etapa de revisar, quando o time acredita que o desenvolvimento da funcionalidade foi concluído e ela está pronta, eles a apresentam para os *stakeholders* (REBELO, 2014).

Para Hendrickson (2011), existem muitas ferramentas que suportam ATDD. Por exemplo, Fit [http://fit.c2.com/] e Robot Framework [http://robotframework.org/]. No entanto, antes de pensar em ferramentas, é importante ter estabelecido uma boa base de processos.

No geral, o ATDD tem algumas características com outra prática bastante popular no mercado, o BDD. Alguns escritores até propõe que se trata da mesma técnica, porém Hammond e Umpress (2012) afirmam que da mesma forma que o BDD é um approach de escrever testes para verificar se as funcionalidades estão corretas, o ATDD é outro approach.

2.6 BDD

O *behavior-driven development* ou desenvolvimento guiado por comportamento, é uma técnica de desenvolvimento ágil focada na colaboração entre desenvolvedores, testadores, analistas de negócios ou mesmo pessoas não técnicas. Segundo North (2006), o idealizador do BDD, a ideia de usar a palavra "*behavior*" no lugar da palavras "*test*" ocasionou uma mudança de pensamento tão profunda, que ele preferiu chamar de *behavior-driven development,* ao invés de continuar usando o termo *test-driven development*.

O BDD foi criado como uma resposta ao *test-driven development* e vem se tornando muito popular nos últimos anos. Por fazer uso de uma linguagem ubíqua, os artefatos (estórias e cenários) são escritos em uma linguagem padrão que facilita a conversação entre toda a equipe envolvida. Assim, os próprios *stakeholders* podem escrever testes (SOLÍS; WANG, 2011). Para Vieira e Pufal (2013), os benefícios de usar uma linguagem que todos entendam vai além do código, é uma questão de aprimorar a comunicação com todos os membros do time.

De acordo com North (2006), o comportamento de uma estória é simplesmente o seu critério de aceitação. Como exemplo, o autor usar um modelo de estórias que possui três elementos considerados essenciais para o desenvolvimento de uma estória. São eles: o usuário que será beneficiado, o que o usuário deseja fazer e qual o beneficio ou valor que essa funcionalidade agrega ao sistema.

As a	e-commerce visitor
I want to	add products to the basket
So that	I can review the items in the shopping cart

Quadro 1: Exemplo de estória

Através do modelo apresentado no Quadro 1, é possível identificar requisitos incompletos ou que não fazem sentido para o sistema. Com a estória definida, os critérios de aceitação podem ser escritos na forma de cenários, que também possuem um modelo de estrutura pré-definido composto por: uma condição inicial, o evento que deve acontecer, e qual a condição esperada.

Scenario: Adding items in the shopping cart	
Given	I am logged in the system with "mary.smith" user and "123@abc" password
And	I am on "cell phones" page
When	I click on the "first deal banner"
And	I click on the "add to cart button"
Then	I have "1" item in the shopping cart

Quadro 2: Exemplo de cenário

Como mostra o Quadro 2, além do *Given*, *When* e *Then*, outros operadores podem usados para conectar os passos. Por exemplo, os operadores *And* e *Or*. North (2006) ainda comenta sobre a importância do reuso de frases nos cenários. Assim, o esforço de implementação é reduzido devido a reutilização de frases já implementadas. Como mostra o Quadro 2, existe uma única implementação para a frase “I click on the <parameter>”. E a mesma frase foi usada duas vezes no mesmo cenário.

As ferramentas utilizadas ao adotar o BDD precisam está conectadas com todas essas características citadas. Algumas ferramentas comumente usadas nos projetos que adotam o BDD são: JBehave (desenvolvido pelo Dan North, o idealizador do BDD) [http://jbehave.org/], Cucumber [https://cucumber.io/], SpecFlow (também conhecido como o Cucumber para .NET) [http://www.specflow.org/], Behat [http://docs.behat.org/en/v2.5/], e outras.

2.7 CONSIDERAÇÕES FINAIS

Neste capitulo apresentou-se toda a fundamentação teórica necessária para o entendimento dos resultados dessa pesquisa. Iniciando com processos de software, passando por metodologias ágeis, qualidade de software, até chegar nas práticas que são objetos de estudo desse trabalho. No capítulo seguinte é apresentado os resultados das entrevistas de acordo com as perguntas de pesquisa.

3 METODOLOGIA

Sobre o método cientifico, Gil (2008, p. 8) considera:

> Para que um conhecimento possa ser considerado científico, torna-se necessário identificar as operações mentais e técnicas que possibilitam a sua verificação. Ou, em outras palavras, determinar o método que possibilitou chegar a esse conhecimento.
>
> Pode-se definir método como caminho para se chegar a determinado fim. E método científico como o conjunto de procedimentos intelectuais e técnicos adotados para se atingir o conhecimento.

Dessa forma, o presente capítulo tem como objetivo apresentar a metodologia de pesquisa utilizada, o instrumento de coleta de dados e as etapas da pesquisa.

3.1 CLASSIFICAÇÃO DE PESQUISA

Gil (2008) classifica as pesquisas sociais de acordo com seus objetivos em três grandes grupos: exploratórias, descritivas e explicativas. As pesquisas exploratórias tem como objetivo proporcionar maior familiaridade com o tema proposto. Já as descritivas, visam descrever as características de determinada população ou fenômeno. Por último, as pesquisas explicativas buscam identificar fatores que determinam ou contribuem para a ocorrência de determinados fenômenos. A presente pesquisa é caracterizada como descritiva, pois apresenta técnicas padronizadas de coleta de dados, um questionário on-line.

Nesse sentido, o delineamento de pesquisa adotado, levantamento de campo ou *survey*, caracteriza-se pelos dados serem obtidos através de um questionário respondido exclusivamente por pessoas relacionadas a indústria de software. De acordo com Babbie (1999, p. 78), o tipo de pesquisa *survey* apresenta semelhanças com outros tipos de pesquisa, em especial, com censos. No entanto, para o autor, “um *survey*, tipicamente, examina uma amostra da população, enquanto o censo geralmente implica uma enumeração da população toda”.

Por fim, na etapa de análise dos dados, abordagens quantitativas e qualitativas foram usadas. Quantitativa porque o questionário disponibilizado apresenta questões objetivas com respostas de sim, não ou talvez sobre o uso das práticas. Nessa abordagem foram usadas análises estatísticas nos dados coletados. A abordagem qualitativa foi usada para entender o uso das práticas utilizadas nas empresas através das questões subjetivas.

3.2 O FORMULÁRIO

Um dos componentes típicos de um *survey* é o questionário, que é aplicado a uma amostra da população. No caso dessa pesquisa, a população entrevistada restringiu-se a pessoas que já trabalharam ou trabalham em projetos de software de empresas atuantes no Brasil e que fazem uso do TDD ou alguma de suas extensões.

O questionário aplicado (ver no Apêndice A) foi divulgado em mídias Sociais (Facebook, LinkedIn e Twitter), listas de e-mails de universidades e através de contato com algumas empresas. Para cada estado brasileiro, pelo menos uma universidade e uma empresa foi contatada, visando uma distribuição mais uniforme das repostas no território brasileiro.

O formulário foi aberto ao público no dia 5 de dezembro de 2015 e aceitou respostas até o dia 10 de janeiro de 2016. Para garantir a privacidade de cada indivíduo, os respondentes não precisavam identificar-se. Assim, era uma opção do entrevistado disponibilizar ou não o e-mail para receber os resultados consolidados. Ao todo, foram 69 respondentes.

No geral, o formulário possuía 15 questões, sendo elas obrigatórios, não obrigatórias, subjetivas e objetivas. Além disso, o formulário foi dividido em três partes. Sendo a primeira, o termo de consentimento livre e esclarecido, que visa oferecer detalhes sobre a participação de cada voluntário.

Após concordar com os termos de consentimento, o usuário é direcionado para a seção dois, que é composta por cinco perguntas. As perguntas dessa seção tem como objetivo fornecer uma visão geral sobre as técnicas utilizadas, a empresa que adotou as práticas e a sua localização e características do projeto. Nessa

seção, segundo os respondentes, a prática mais utilizada foi o TDD, seguido logo após pelo BDD, como mostra o Gráfico 1.

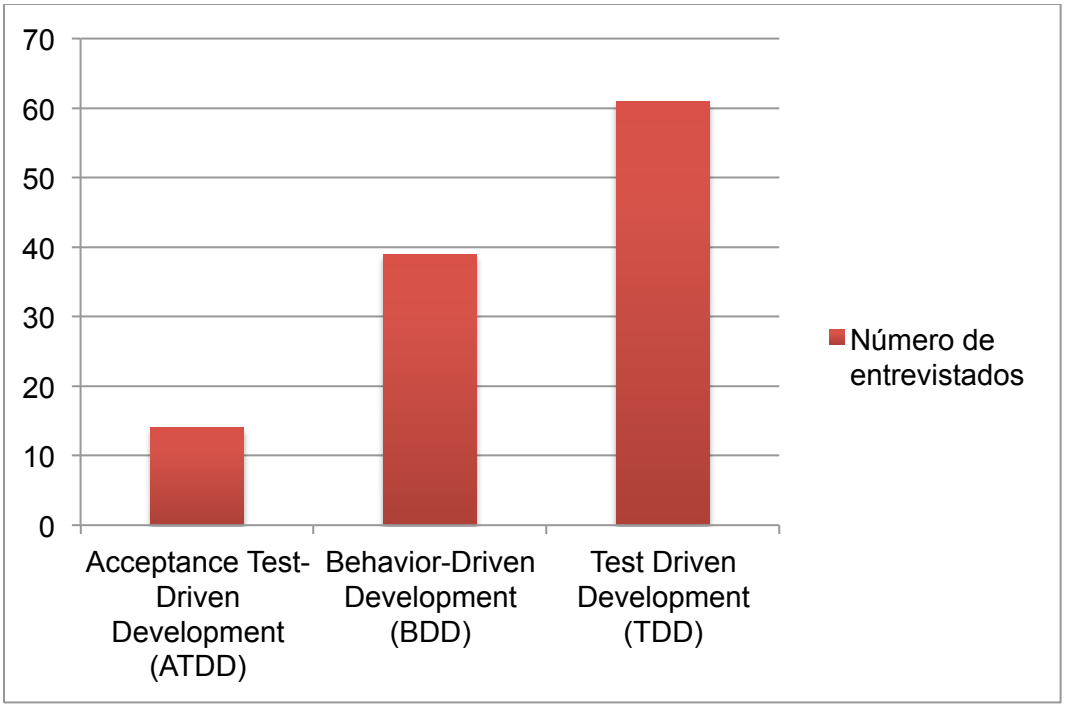

Gráfico 1: Práticas usadas pelos entrevistados

A única restrição com relação aos respondentes era com relação ao uso das técnicas. A pessoa já deveria ter usado pelo menos uma das práticas em empresas localizadas no Brasil.

Por último, se o entrevistado já tivesse trabalhado ou trabalha com alguma das técnicas apresentadas, ele ou ela era direcionado para a seção 3. Essa seção está relacionada a experiência do usuário com as técnicas citadas na seção 2. Existem perguntas objetivas e subjetivas nessa seção relacionados aos benefícios, dificuldades e mudanças identificadas nos projetos.

3.3 ETAPAS DE PESQUISA

Para atingir os objetivos de pesquisa, o desenvolvimento do trabalho foi dividido nas seguintes etapas:

Revisão da literatura: nessa etapa foram levantadas as principais pesquisas sobre desenvolvimento de software, qualidade e práticas usadas na engenharia de software.

Análise dos dados coletados: os dados coletados da amostra foram analisados, através do Microsoft Excel e do Formulários Google [https://docs.google.com/forms/]. Essas ferramentas foram utilizadas com o intuito de facilitar o agrupamento e interpretação dos dados.

Resultados alcançados: apresentação dos resultados encontrados na pesquisa.

3.2 CONSIDERAÇOES FINAIS

Neste capítulo foi abordado toda a metodologia de pesquisa usada no presente trabalho. Desde a forma como a pesquisa foi classificada e distribuída até como a abordagem utilizada na análise dos dados. O próximo capítulo apresenta a análise dos resultados coletados na pesquisa.

4 ANALISE DOS RESULTADOS

O presente capítulo apresenta os resultados encontrados na análise dos dados coletados no formulário online com relação às dificuldades, benefícios e mudanças enfrentadas pelas empresas brasileiras durante e após a adoção do TDD e suas extensões.

4.1 CARACTERÍSTICAS GERAIS DA AMOSTRA

A segunda seção do questionário buscou informações sobre as práticas usadas pelos entrevistados, o perfil das empresas que eles trabalham ou trabalhavam, o tamanho dos times e em qual a função dos entrevistados na organização. Com exceção da primeira pergunta que era relacionada as práticas utilizadas pelos entrevistados (Gráfico 1), as demais perguntas eram subjetivas e o entrevistado poderia fazer associações com a técnica previamente citada.

4.1.1 Localização Geográfica

A partir da pergunta dois da seção dois (*Você trabalhava em qual estado quando utilizou a(s) técnica(s) citada(s) na pergunta 1?*) foi possível identificar o estado brasileiro, onde o respondente que utilizou a técnica trabalha. Juntamente com a pergunta um da seção dois (*Você já utilizou alguma das técnicas abaixo? ATDD, BDD, TDD, ou Outros*), pode-se realizar o mapeamento de técnicas utilizadas por estado, conforme o Gráfico 2.

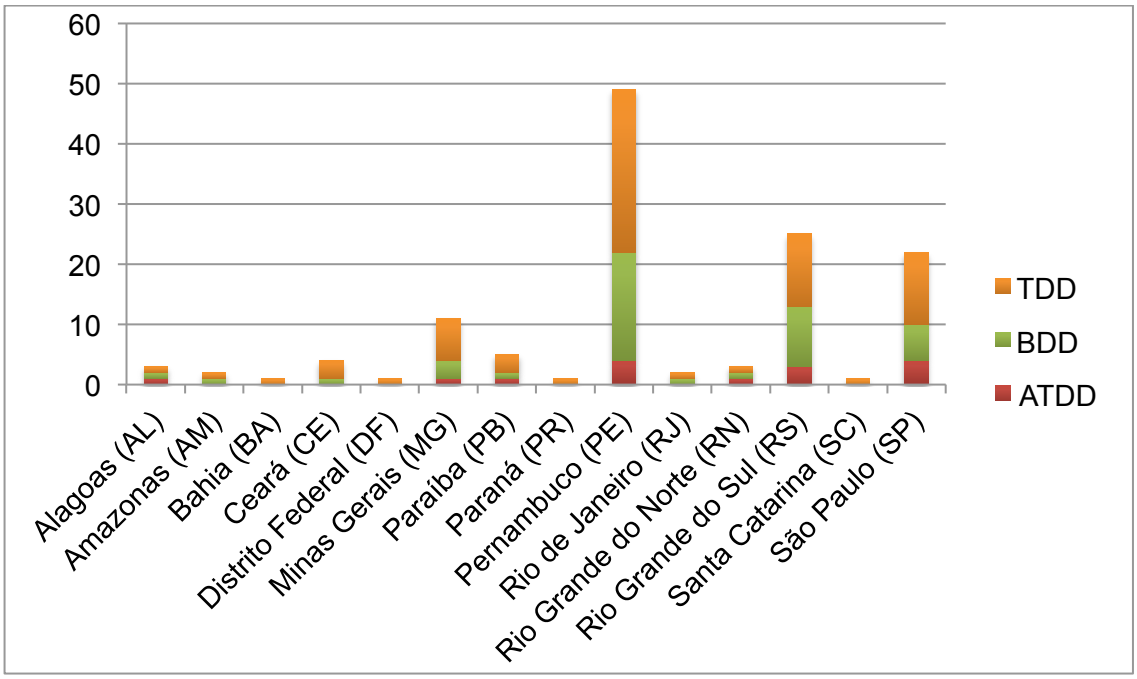

Gráfico 2: Práticas usadas por estado brasileiro

Como mostra o Gráfico 2, a maioria dos respondentes eram de empresas localizadas em Pernambuco, Rio Grande do Sul e São Paulo. Treze estados não tiveram representação na pesquisa. Embora, pelo menos uma empresa e uma universidade de cada estado foi contatada para responder a pesquisa. No geral, TDD foi a prática mais utilizada em todos os estados.

4.1.2 As Empresas

Ao todo, responderam a pesquisa participantes de diferentes perfis, que já participaram do processo de desenvolvimento de software das mais variadas empresas atuantes no território brasileiro. Foram identificadas 33 empresas distintas na pergunta três da seção dois (*Onde você trabalhava quando utilizou a(s) técnica(s) citada(s) na pergunta 1?*). Entre elas multinacionais, universidades, órgãos do governo e pequenas empresas.

Na sua maioria, as empresas eram do ramo de Tecnologia da Informação (TI), onde os produtos finais são soluções tecnológicas. O Gráfico 3 mostra a percentagem das empresas de TI coletadas na amostra. As empresas que foram classificadas como de outros ramos, podem possuir uma equipe interna para

atender as necessidades ou já ter usado a prática em conjunto com outra empresa. Por exemplo, uma empresa terceirizada.

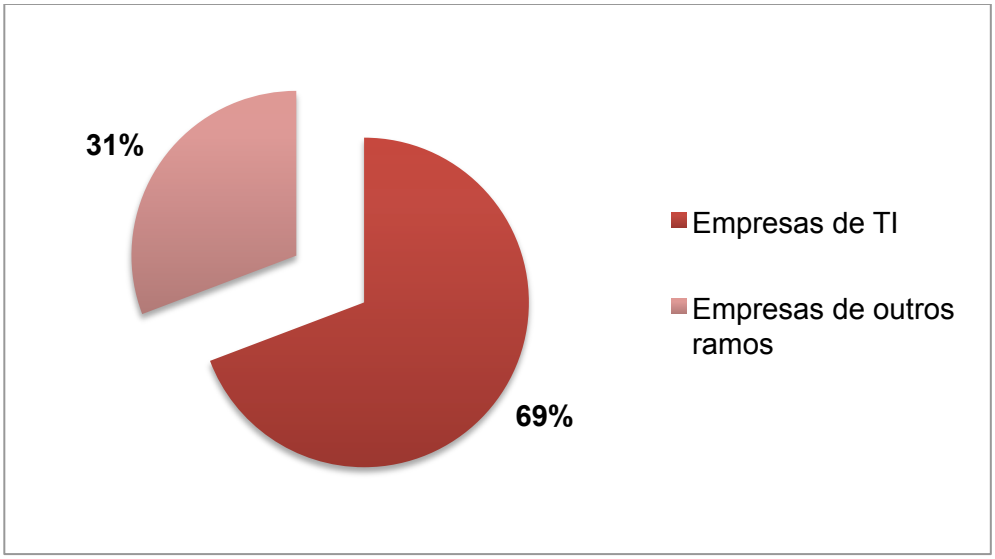

Gráfico 3: Ramo das empresas coletadas na amostra

4.1.3 Papéis desempenhados pelos entrevistados

A pergunta quatro da seção dois (*Qual o papel que você desempenhava no projeto quando utilizou a(s) técnica(s) citada(s) na pergunta 1 ?*) tinha como objetivo entender o perfil dos entrevistados, com relação aos papéis desempenhados por eles nas organizações. Como a única restrição relacionada aos respondentes era a respeito do uso das técnicas, os perfis dos respondentes foram desde *product owner* a engenheiro de testes.

O Gráfico 4 demonstra o perfil dos respondentes de forma uniformizada. Ou seja, os perfis de analista de teste, testador e engenheiro de teste foram agrupados no mesmo grupo chamado “Engenheiro de Teste”. O mesmo aconteceu para os respondentes que informaram como papel ou perfil no projeto os cargos de desenvolvedor, engenheiro de sistemas e engenheiro de software. Eles foram agrupados no perfil chamado “Desenvolvedor”.

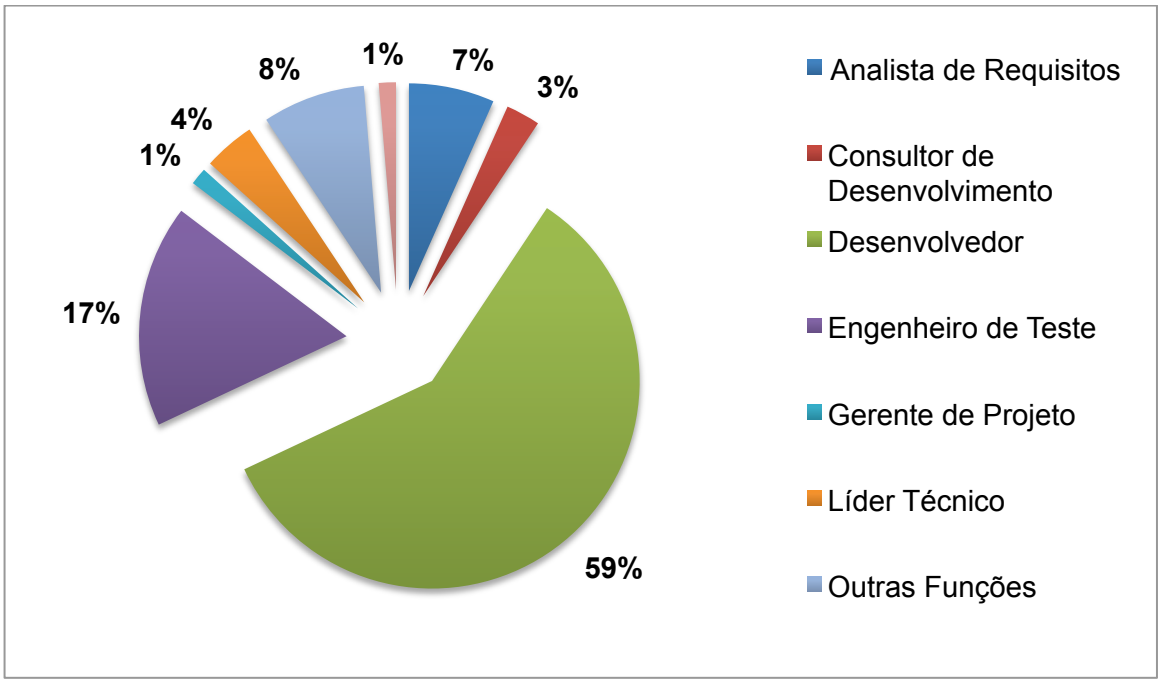

Gráfico 4: Papel desempenhado pelos entrevistados

Dessa forma, o gráfico 4 mostra que mais da metade da amostra, trabalhava como desenvolvedor nos projetos, cerca de 59%. Logo em seguida, ficaram os engenheiros de testes que representam 17% da amostra. É importante ressaltar que no formulário, as pessoas poderiam ter trabalhado com mais de um cargo. Isso também foi levado em consideração nessa contagem. Por exemplo, se um entrevistado trabalhou como desenvolvedor com uma técnica, e como analista de requisitos com outra. As duas funções para esse entrevistado foram consideradas.

4.1.4 Tamanho dos Times

A literatura prega que times ágeis geralmente são pequenos. Em particular, Faria (2013) comenta que o Scrum prega que os times tenham entre 5 e 9 pessoas. No entanto, o resultado da pergunta cinco da seção dois (*Quantas pessoas (mais ou menos) trabalhavam no projeto quando você utilizou a(s) técnica(s) citada(s) na pergunta 1?*) mostra que o tamanho dos times variava bastante e muitas vezes fugiu da regra de times pequenos (Gráfico 5).

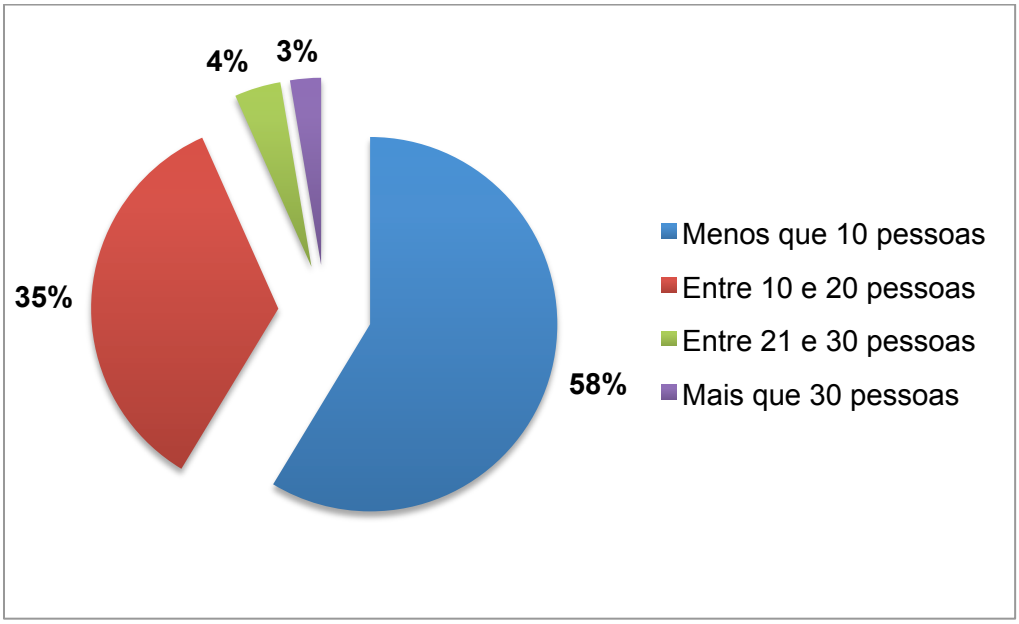

Gráfico 5: Tamanho dos times

O Gráfico 5 mostra que os times em sua maioria, tinham menos de 21 pessoas por time. Por exemplo, 58% dos entrevistados afirmaram que trabalhavam em times que tinham menos de 10 pessoas. Inclusive, 2 desses entrevistados afirmaram que trabalhavam sozinho no projeto.

4.2 TDD E EXTENSÕES NO CONTEXTO BRASILEIRO

A seção três do questionário on-line buscou informações sobre dificuldades, benefícios e mudanças enfrentadas por empresas brasileiras ao adotarem as práticas citadas nesse estudo.

Cada item, dificuldade, beneficio, ou mudança, era representado no formulário por duas perguntas. A primeira, uma pergunta objetiva com possíveis respostas: sim, não e talvez. Cujo o intuito consistia na identificação quantitativa da prática. Posteriormente, uma segunda pergunta subjetiva visava entender o reflexo do uso das práticas na empresa. As subseções abaixo estão relacionadas respectivamente as dificuldades, benefícios e mudanças encontradas nos dados coletados.

4.2.1 Dificuldades no processo de adoção

A partir da pergunta um da seção três do formulário (*Seu time encontrou dificuldades na adoção da(s) prática(s) escolhida(s)?*) foi identificado que, 54% dos times tiveram dificuldade na adoção de alguma prática citada neste trabalho.

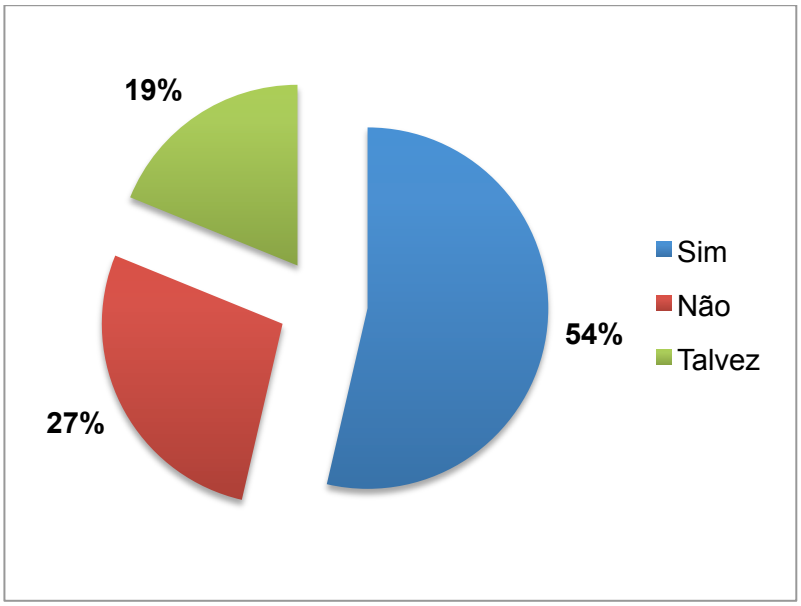

Gráfico 6: Percentual de entrevistados que encontraram dificuldades no processo de adoção.

Destes 54% (37 entrevistados) que sentiram dificuldades ao adotar o TDD ou suas extensões, através do Gráfico 7, podemos identificar que 8 usavam ATDD, 17 usavam BDD e 34 usavam TDD.

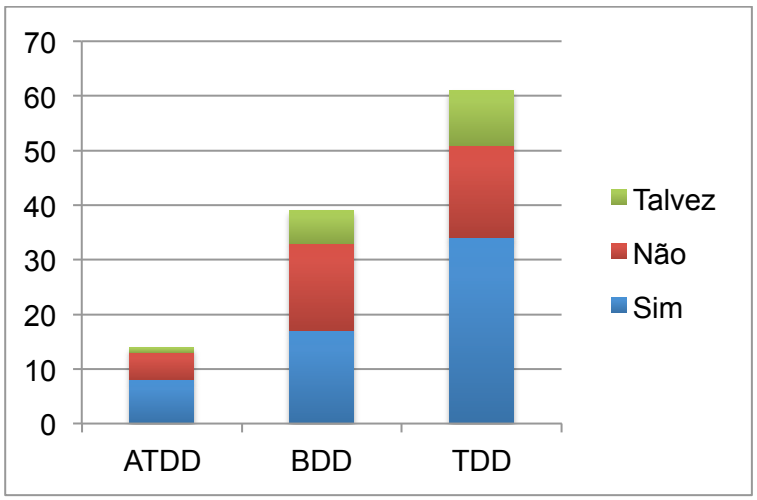

Gráfico 7: Entrevistados que encontraram dificuldades no adoção versus técnica adotada.

No entanto, visando entender quais foram essas dificuldades que a maioria dos entrevistados enfrentou, foi formulada a pergunta dois da seção três (*Para cada prática usada, quais foram as principais dificuldades que o seu time encontrou no processo de adoção?*). Através dessa pergunta, descobriu-se que as maiores dificuldades enfrentadas pelo uso do TDD foram:

- Falta de conhecimento da técnica: apesar de existir muitos livros e artigos sobre TDD, engenheiros de testes e desenvolvedores afirmaram que no começo da adoção do TDD, uma das maiores dificuldades era o conhecimento técnico e a falta de exemplos reais de como usá-la.

- Conscientização e resistência da equipe: essa, sem dúvida, foi a principal dificuldade citada pelos entrevistados. Segundo Cardoso (2013), a causa dessa resistência se deve ao fato de: “o código que realizará sua lógica deve ser criado somente após a criação do teste e isso torna-se algo de difícil aceitação pois ainda não se tem nada e já se faz necessário testar”. Por isso, Hammond e Umpress (2012) comentam que o uso do TDD requer pessoas com experiência, talentosas e flexíveis.

- *Mockups* complexos: para desenvolvedores, o uso de *mockups* deixa o processo muito complexo. De acordo com Beck (2002), o uso de *mocks* ajuda na redução de acoplamento entre os objetos. Através de *mocks*, os usuários conseguem representar o comportamento da classe, mesmo antes de codificá-la.

- Dificuldade em fazer a rastreabilidade: alguns desenvolvedores comentaram que tiveram dificuldades para fazer o rastreamento entre classes, testes e requisitos.

O total de entrevistados que são usuários do BDD, e que responderam a pergunta dois da seção 3 foram 20. Segundo os dados coletados, as maiores dificuldades enfrentadas pelos time são:

- Falta de conhecimento da técnica e de ferramentas: semelhantemente ao TDD, os adeptos do BDD, também informaram que tiveram dificuldades com o conhecimento de tecnologias usadas com o BDD e da própria técnica.

- Desenvolvimento dos dicionários: para alguns engenheiros de testes entrevistados na pesquisa, a definição e implementação dos dicionários foi uma das maiores dificuldades enfrentadas pelo time. North (2006), defende o reuso de frases o máximo possível para diminuir esse esforço de implementação.

- Mudança de pensamento: engenheiros de testes e desenvolvedores concordam com Hammond e Umpress (2012), quando ele afirma que, o BDD propõe uma mudança de foco. Para os entrevistados, essa mudança é muito difícil e alguns membros da equipe ofereceram resistência a mudança de pensamento.

- Importância para o cliente: para um dos entrevistados, o cliente não considerava a técnica importante e não colaborava com os artefatos do BDD. Segundo ele, o "cliente não considerava as especificações como um documento de negócios servindo apenas para o time de dev/testes/analistas gerando retrabalho por ter que escrever outros documentos" .

Por último, os usuários do ATDD, mesmo em menor número, destacaram algumas das dificuldades encontradas por eles no processo de adoção. A falta de conhecimento pela equipe sobre a técnica e de ferramentas também foi mencionado pelos entrevistados como um entrave inicial. As principais dificuldades encontradas foram

- Falta de conhecimento da técnica e de ferramentas: semelhante as outras técnicas, os usuários do ATDD também destacaram a falta de conhecimento como uma das maiores dificuldades no processo de adoção.

- Produtividade *versus* vantagens: a adoção do ATDD tem uma curva de aprendizado um pouco longa devido as mudanças no processo de desenvolvimento. Algumas vezes, os desenvolvedores não enxergam as vantagens do uso da técnica e achavam que o time estava perdendo produtividade sem nenhum ganho.

- Comunicação: o ATDD propõe um ciclo de desenvolvimento colaborativo composto por 4 etapas, entre elas uma etapa de discussão dos requisitos

(HENDRICKSON, [2008]). No entanto, um dos entrevistados informou que nem sempre é fácil entrar em contato com as pessoas do negócio, o que dificulta bastante o processo.

Em resumo, a Tabela 1, apresenta as três principais dificuldades enfrentadas pelos entrevistados na adoção de cada técnica.

Técnica	**Dificuldades**	**Número de entrevistados**
ATDD	Falta de conhecimento da técnica e ferramentas	4/9
ATDD	Produtividade *versus* vantagens	4/9
ATDD	Comunicação	1/9
BDD	Falta de conhecimento da técnica e de ferramentas	8/20
BDD	Mudança de pensamento	6/20
BDD	Desenvolvimento dos dicionários	3/20
TDD	Conscientização e resistência da equipe	24/42
TDD	Falta de conhecimento da técnica e ferramentas	14/42
TDD	*Mockups* complexos	4/42

Tabela 1: Resumo das dificuldades

Através da Tabela 1, é possível observar quantas vezes uma dificuldade foi citada pelas pessoas que usavam a técnica e responderam a pergunta subjetiva relacionada as dificuldades. Por exemplo, para o BDD, 39 pessoas usavam a técnica, mas apenas 20 listaram dificuldades (pergunta dois da seção dois).

4.2.2 Benefícios a curto e longo prazo identificados

Através pergunta três da seção três (*Foram identificados benefícios após a adoção?*), pode-se concluir que colaboradores de empresas brasileiras acreditam sim nas técnicas apresentadas nesse trabalho. Dos 69 selecionados na pesquisa, 58 afirmaram ter identificado algum beneficio após a adoção do TDD ou suas

extensões. Apenas 2 pessoas disseram que não encontraram benefícios e 9 informaram que talvez alguns benefícios foram identificados (Gráfico 8).

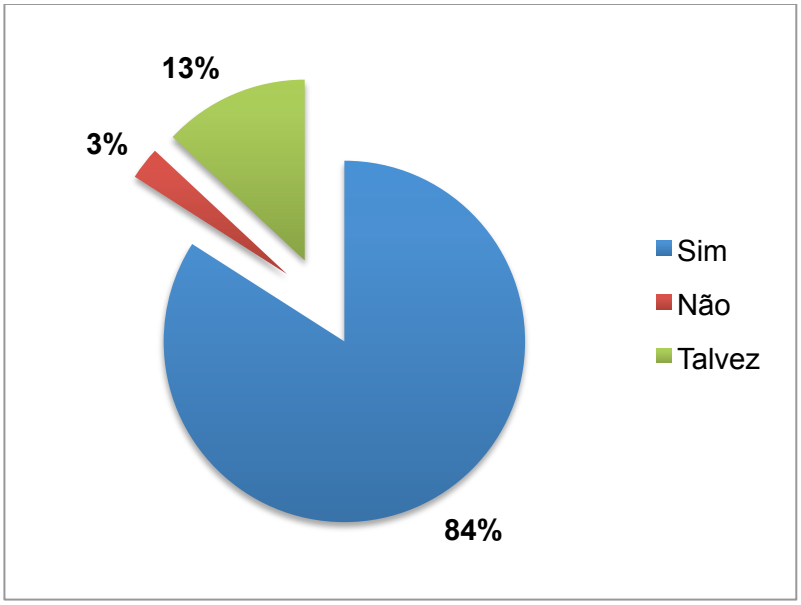

Gráfico 8: Percentual de entrevistados que identificaram benefícios após a adoção

De forma bastante positiva, o Gráfico 8 mostra que 84% dos entrevistados enxergam benefícios com o uso do ATDD, BDD ou TDD. Mais detalhadamente, o Gráfico 9 mostra a resposta de cada entrevistado de acordo com as técnicas utilizadas.

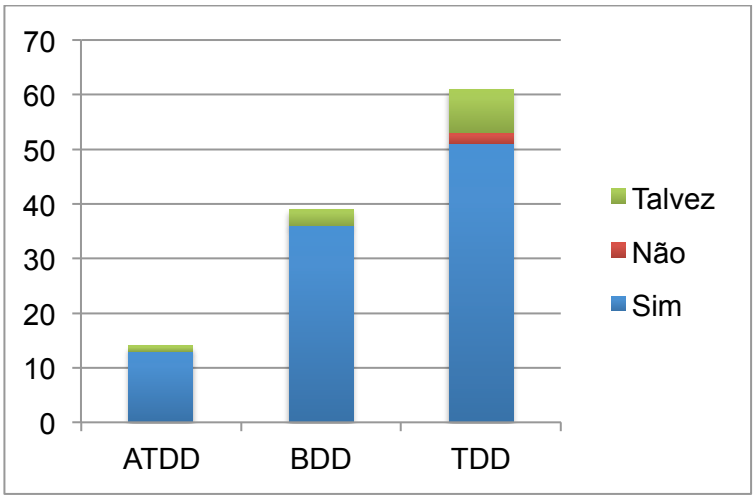

Gráfico 9: Entrevistados que identificaram benefícios após a adoção versus técnica adotada.

Por exemplo, através do Gráfico 9 é possível identificar a técnica usada pelas duas pessoas que informaram que não identificaram benefícios após a adoção das práticas. No caso, elas usavam TDD. Além disso é possível visualizar a quantidade de respostas “Sim”, “Não” e “Talvez” para técnica.

De forma mais subjetiva, a pergunta quatro da seção 3 (*Para cada prática usada, quais foram os benefícios a curto e longo prazo identificados?*) buscou identificar esses benéficos. Devido a variedade de times e empresas dos entrevistas, alguns benefícios são bem particulares. No entanto, outros podem ser considerados mais genéricos, visto que, outros entrevistados apresentaram as mesmas ideias.

Segundo os entrevistados que afirmaram usar TDD, os benefícios mais citados foram:

- Qualidade do código: como boa parte dos entrevistados são desenvolvedores, esse foi o beneficio mais citado. Para eles, através do TDD, é possível escrever um código enxuto e com um *design* muito bem feito. Sob o mesmo ponto de vista, Beck (2002, p. 15) declara que o objetivo do TDD é: “*clean code that works”*.

- Redução do número de *bugs*: esse beneficio foi notado por diferentes papeis, desde desenvolvedores a engenheiros de teste. De acordo com esses profissionais, é notável a diminuição dos *bugs* encontrados. Um engenheiro de testes da amostra informou que “durante o desenvolvimento podemos perceber que a quantidade e criticidade dos bugs que ainda eram descobertos foram baixíssimas”.

- Velocidade para obter *feedbacks*: para os desenvolvedores, durante o desenvolvimento e após a conclusão de uma estória é importante saber se a situação da mesma. Por isso, desenvolvedores consideram o TDD muito útil nesse quesito de oferecer um feedback rápido sobre determinada estória.

- Cobertura da suíte: para Beck (2002), se o TDD for seguido rigorosamente, o nível de cobertura deve ser 100%. Por isso, que desenvolvedores, engenheiros de testes e até um gerente de projeto, levantaram esse como mais um dos benefícios do TDD.

- Facilidade para rodar regressões: a facilidade e rapidez na execução de uma regressão do sistema foi considerado outro beneficio do uso do TDD. O mesmo beneficio foi citado por gerente de projetos, desenvolvedores e engenheiros de testes.

- Segurança para entregas: esse beneficio foi citado, em sua maioria, por *product manager*, líder de equipe e consultor de desenvolvimento. Para eles, era notável a segurança da equipe em realizar uma entrega ou homologação para o cliente. Cardoso (2013) afirma que, essa segurança se deve ao fato que os desenvolvedores confiam, pois sabem que cada trecho do *software* foi devidamente testado.

Um dos benefícios do TDD, a cobertura da suíte de testes, apresentou resultados bastante semelhantes a outra prática, o BDD. São eles:

- Cobertura da suíte: esse benefício assemelhou-se muito ao apresentado pelo TDD.

- Melhoria no entendimento do negócio: Esse beneficio foi praticamente unânime entre os usuários do BDD, principalmente para engenheiro de testes e analistas de requisitos. Hammond e Umpress (2012) estavam certos quando afirmaram que, o BDD propõe uma mudança de foco para o desenvolvedor. Ele ou ela deixa de pensar e código e começa a pensar na funcionalidade em si.

- Facilidade para escrever cenários: para os engenheiros de testes, um beneficio muito citado foi a facilidade para escrever cenários. Para North (2006), essa facilidade se deve ao fato que, o BDD usa um vocabulário em uma linguagem ubíqua, onde até mesmo pessoas não técnicas conseguem ler.

- Facilidade de discutir os requisitos: especialmente para um consultor de desenvolvimento, o uso de uma linguagem ubíqua também facilita a discussão entre as pessoas do negócio. Até mesmo o fluxo da aplicação fica mais fácil validar com os *stakeholders* com a ajuda do BDD. Para Vieira e Pufal (2013), essa facilidade se deve ao fato de “ter os analistas de negócio

dialogando com os desenvolvedores e analistas de qualidade, todos eles aprimorando aquele único arquivo que é uma maneira não abstrata de demonstrar ideias".

- Velocidade após a implementação das principais frases: como comentado por North (2006), o reuso das frases deve ser incentivado para que o esforço de implementação seja diminuído. Para um dos entrevistados, "o reuso de *steps* em comum diminuiu tempo de desenvolvimento de código de testes, permitindo a criação de mais cenários".

Já o ATDD foi a prática menos popular entre os entrevistados. Apenas 14 deles utilizam ou já utilizaram a prática (Gráfico 1). Por isso, os benefícios apresentados para o ATDD talvez não sejam tão gerais como os demais. Em síntese, esses são os benefícios do ATDD mais genéricos coletados na amostra:

- Confiança e entendimento claro dos requisitos: para os usuários do ATDD, o debate dos requisitos é o ponto inicial do ciclo ATDD. O fato dos clientes participarem da discussão dos requisitos enriquece o processo. De acordo com Hammond e Umpress (2012), no ATDD, os clientes inserem dados em uma tabela usando aplicativos de texto ou planilhas, e os desenvolvedores podem usa-los em seus testes. Para eles, essa colaboração contribui com o esclarecimento dos requisitos.

- Qualidade do produto: desenvolvedores, líderes de equipe e arquitetos, acreditam que a qualidade do produto é um dos maiores benefícios dos usuários. Esse foi um dos tópicos mais lembrados pelos entrevistados. Hammond e Umpress (2012) acreditam que isso é possível porque engenheiros de *software* passam a focar em testes de alto nível para obter um contexto mais geral da aplicação.

- Facilidade de encontrar mudanças que afetam determinados grupos de usuários: para os desenvolvedores, através do ATDD, é fácil identificar as mudanças que afetam os usuários, devido ao desacoplamento das estórias.

- Segurança para manutenção e alterações de código: esse benefício está diretamente relacionado ao anterior. Os desenvolvedores que usavam ATDD

informaram que se sentiam seguros para fazer alterações no código, pois sabiam que aquela alteração não iria quebrar outros pontos.

Mediante o exposto, a Tabela 2 mostra de forma resumida os benefícios mais citados para cada técnica e por quantos entrevistados cada tópico foi citado.

Técnica	**Benefício**	**Número de entrevistados**
ATDD	Qualidade do produto	4/10
ATDD	Segurança para manutenção e alterações de código	3/10
ATDD	Conhecimento e entendimento claro dos requisitos	2/10
BDD	Facilidade para escrever cenários	6/26
BDD	Melhoria no entendimento do negócio	5/26
BDD	Velocidade após a implementação das frases	5/26
BDD	Cobertura da suíte	4/26
TDD	Qualidade do código	29/35
TDD	Segurança para entregas	13/35
TDD	Cobertura da suíte	7/35
TDD	Facilidade para rodar regressões	7/35

Tabela 2: Resumo dos benefícios

Como a pergunta de identificação dos benefícios não era obrigatória, nem odos os entrevistados a responderam. Ao todo 54 pessoas responderam a pergunta quatro da seção três. Entre os 54 respondentes, 10 usavam ATDD, 26 usavam BDD e 35 usavam TDD. É importante lembrar que o entrevistado pode ter listado benefícios para mais de uma técnica na pergunta quatro da seção três, por isso a resposta dele vai ser contabilizada na múltiplas técnicas que ele citou. É importante ressaltar que a maioria dos benefícios identificados são de teor técnico, visto que 76% dos entrevistados eram desenvolvedores ou engenheiros de testes (Gráfico 4).

4.2.3 Mudanças identificadas

A pergunta cinco da seção três (*Foram identificadas mudanças no projeto/time/empresa após a adoção?*), demonstrou que sim, em sua maioria, as pessoas detectaram mudanças após a adoção de qualquer uma das práticas (Gráfico 10).

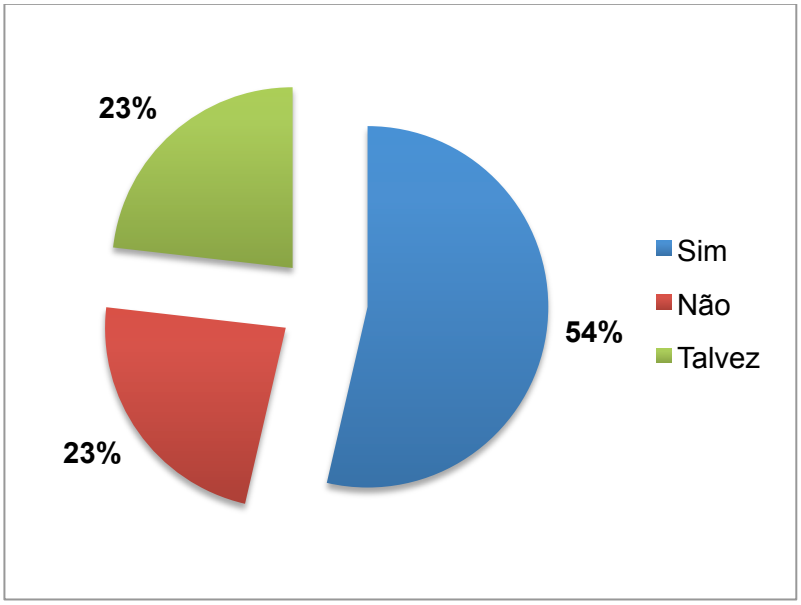

Gráfico 10: Percentual de entrevistados que identificaram mudanças no projeto, time ou empresa após a adoção.

O Gráfico 10 mostra que mais de 50% dos entrevistados detectaram mudanças, esse percentual também se manteve na mesma média entre as técnicas (Gráfico 11) .

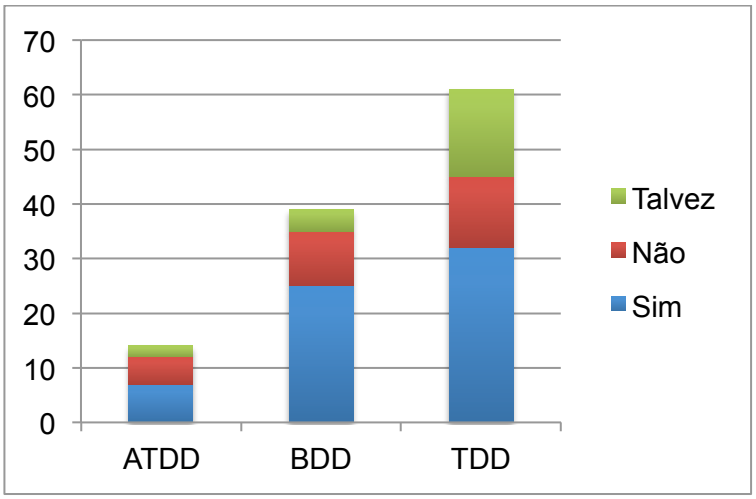

Gráfico 11: Entrevistados que identificaram mudanças após o processo adoção versus técnica adotada.

Apesar da maioria dos entrevistados afirmarem que identificaram mudanças após a adoção do ATDD, TDD ou BDD, as respostas coletadas pelos entrevistados na pergunta seis da seção três (*Para cada prática usada, quais foram as principais mudanças identificadas?*) eram muito relacionadas ao benefícios ou dificuldades. No geral, as mudanças identificadas após o uso do TDD foram:

- Mudança de pensamento de como fazer as coisas: para os desenvolvedores entrevistados, a maior mudança está relacionada a pensar no resultado final antes mesmo de começar a fazer. Sob o mesmo ponto de vista, Hammond e Umpress (2012) afirmam que, o TDD pode ser usado com approach mais estruturado, no entanto sempre pensando com o objetivo final.

- Interação entre as equipes: de acordo com os entrevistados, o uso do TDD provocou mudanças na forma como as equipes de desenvolvimento interagiam com os demais. Para eles, após a técnica, a equipe passou a interagir muito melhor com os *stakeholders* e demais membros do projeto.

- Mudança de cultura: a mudança de cultura está muito relacionada ao primeiro item dessa lista, mudança de pensamento. No entanto, o foco aqui é muito mais geral. Por exemplo, segundo um entrevistado, até o que a equipe buscava nos novos contratados foi redefinido. Essas mudanças foram feitas para buscar pessoas com mais experiência com TDD que possa agregar

muito valor ao time, e para encontrar pessoas que mesmo com pouca experiência, mas estejam disponíveis para aprender e somar ao time.

Dois dos entrevistados acharam as mudanças causadas pelo BDD tão impactantes, que resolveram levar a experiência para outros projetos. No geral, para os adeptos do BDD, as mudanças mais citadas foram:

- Engajamento da equipe: desenvolvedores e engenheiros de testes comentaram sobre mudanças no comportamento da equipe. Para eles, a equipe ficou mais motivada a entender as funcionalidades das solução proposta e aumentou o interesse em contribuir para um projeto melhor.

- Processo de comunicação: dois entrevistados relataram mudanças no processo de comunicação. Para eles, a comunicação ficou mais eficiente após o uso do BDD. Segundo Vieira e Pufal (2013), a comunicação deve sim ser aprimorada entre todos os membros do time.

Infelizmente, entre os adeptos do ATDD, nenhum dos sete entrevistados que respondeu “sim” a pergunta cinco da seção três (Gráfico 11), respondeu a questão seis da mesma seção. Um entrevistado, que respondeu “Não” na pergunta cinco da seção três, explicou a situação. Segundo ele, o motivo foi a falta de apoio da diretoria que, acabou não ajudando na adoção das técnicas. A tabela 3 apresenta um resumo das principais mudanças encontradas nos dados coletados.

Técnica	**Mudanças**	**Número de entrevistados**
BDD	Engajamento da equipe	4/14
BDD	Processo de comunicação	2/14
TDD	Mudança de pensamento em como fazer as coisas	6/23
TDD	Interação entre as equipes	3/23
TDD	Mudança de cultura	3/23

Tabela 3: Resumo dos mudanças identificadas

No geral, apenas 29 pessoas responderam a pergunta subjetiva relacionada aos desafios. Além disso, algumas dessas respostas não eram necessariamente mudanças. Estavam mais relacionadas as outras perguntas de benefícios ou dificuldades. Por isso, a quantidade de mudanças identificadas foi mais resumida.

4.3 CONSIDERAÇÕES FINAIS

Através da analise do dados coletados, foi possível concluir que as empresas e profissionais da indústria de software brasileira passam por dificuldades, benefícios e mudanças no processo de adoção do ATDD, BDD e TDD. No entanto, as empresas consegue perceber o valor que essas técnicas agrega ao produto. O capítulo seguinte encerra a pesquisa, apresentando as conclusões, limitações e propostas para trabalhos futuros.

5 CONCLUSÃO

A presente pesquisa buscou entender um pouco do contexto do TDD e algumas de suas extensões na indústria de software brasileira. Através de um formulário disponibilizado por um período um pouco maior do que 30 dias, profissionais da indústria de software que já tivessem trabalhado com o ATDD, BDD ou TDD eram convidados a responder algumas perguntas.

Através da pesquisa, verificou-se que, o TDD é a prática mais popular em todos os estados brasileiros. Dos 69 entrevistados, apenas 8 (aproximadamente 12%) nunca tinham trabalhado com o TDD. Além disso, foi possível identificar que o ATDD ainda é pouco usado no território brasileiro. Poucas entrevistados comentaram sobre o uso da técnica.

Além das dificuldades, benefícios e mudanças já conhecidos pela literatura, através da analise dos dados coletados, foi possível validar os fatores já conhecidos na indústria brasileira e identificar outros fatores relacionados a esses três pilares.

Primeiro, o levantamento dos principais artigos, livros e outros materiais sobre as técnicas ajudou na coleta de dificuldades, benefícios e mudanças comentados mundialmente pela academia e indústria de software. Os dados coletados do questionário serviram para validar o uso do TDD e suas extensões pelos profissionais brasileiros. Sempre procurando comparar com informações previamente coletadas da literatura. Foi possível validar que tanto os dados coletados da pesquisa, como os citados no referencial teórico eram bem parecidos. Ou seja, os profissionais brasileiros estão usando as técnicas de acordo com o que a academia e indústria pregam.

Por último, foi possível identificar dificuldades, benefícios e mudanças que não tinham sido comentadas pela literatura. Do ponto de vista das dificuldades, as mais lembrados pelos entrevistados foram: ATDD - falta de conhecimento da técnica e ferramentas, produtividade versus vantagens e comunicação. BDD - falta de conhecimento da técnica e de ferramentas, mudança de pensamento e desenvolvimento dos dicionários. TDD - conscientização e resistência da equipe, falta de conhecimento da técnica e ferramentas e *mockups* complexos.

No entanto, a pesquisa mostrou que mesmo com dificuldade, mais de 80% dos entrevistados encontraram benefícios com o uso do ATDD, TDD, ou BDD. Os benefícios mais citados foram: ATDD - conhecimento e entendimento claro dos requisitos, qualidade do produto e segurança para manutenção e alterações de código. BDD - cobertura da suíte, melhoria no entendimento do negócio, facilidade para escrever cenários e velocidade após a implementação das frases. TDD - qualidade do código, segurança para entregas, cobertura da suíte e facilidade para rodar regressões.

Por último, uma das questões mais difíceis buscava identificar mudanças na empresa do ponto de vista de processo, pessoas e organização. As mudanças citadas pelos entrevistados foram: BDD - engajamento da equipe e processo de comunicação. TDD - mudança de pensamento em como fazer as coisas, interação entre as equipes e mudança de cultura.

5.1 LIMITAÇÕES E TRABALHOS FUTUROS

Após o levantamento da revisão bibliográfica, verificou-se que algumas técnicas citadas na pesquisa não poderiam ser consideradas extensões do TDD ou existia muito conflito na literatura sobre o tema. Por isso, o *Design by Contract*, o *Story Test Driven Development* e *Specification by Example* não foram levados em consideração no trabalho. No caso do STDD existia muito conflito na literatura sobre a origem e objetivos. Por isso, um estudo aprofundado e mais direcionado é recomendado para clarear esse tema.

Além disso, não foi possível fazer uma avaliação profunda sobre técnica utilizada *versus* estado, pois os maiores números de entrevistados estavam presentes em Pernambuco, Rio Grande do Sul e São Paulo. Para próximas pesquisas, recomenda-se entrar em contato diretamente com mais empresas de todos os estados brasileiros. Seria interessante também ter o apoio de algum órgão de apoio a tecnologia para ajudar na divulgação da pesquisa.

Para finalizar, o uso de perguntas subjetivas enriquece a pesquisa, pois oferece ao entrevistado espaço para o mesmo escrever o que pensa. No entanto, esse método é muito custoso para a interpretação e agrupamento dos dados.

Devido a restrições de tempo, não foi fazer esse estudo prévio para identificar as respostas pré-definidas nesse trabalho. Como trabalho futuro sugere-se, uma pesquisa que disponha de respostas pré-determinadas para facilitar o processo de análise dos dados.

6 REFERÊNCIAS

AGILE ALLIANCE. **Guide to Agile Practices.** Disponível em: <http://guide.agilealliance.org/ >. Acesso em: 12 jan. 2016.

AMBLER, Scott W.. **Introduction to Test Driven Development (TDD)**. 2013. Disponível em: <http://www.agiledata.org/essays/tdd.html>. Acesso em: 30 dez. 2015.

BABBIE, Earl. **Métodos de Pesquisas de Survey.** Belo Horizonte: UFMG, 1999. 78 p. Tradução de Guilherme Cezarino.

BECK, Kent. **Test-Driven Development:** By Example. [S. l.]: Addison-wesley Professional, 2002.

BECK, K. S. K. S. J. E. A.; **Manifesto for agile software development**. 2001. Disponível em: <http://agilemanifesto.org/>. Acesso em: 30 dez. 2015.

CARDOSO, Andre. **TDD, por que usar?** 2013. Disponível em: <http://tableless.com.br/tdd-por-que-usar/>. Acesso em: 15 jan. 2016.

COCKBURN, Alistair. **Crystal Clear:** A Human-Powered Methodology for Small Teams**.** [S. l.]: Addison-Wesley Professional, 2004. 336 p.

HENDRICKSON, Elisabeth. **The ATDD Arch.** 2011. Disponível em: <http://testobsessed.com/2011/02/the-atdd-arch/>. Acesso em: 13 jan. 2016.

International Standards Organization. **ISO 9000:2005**: Quality Management Systems – Fundamentals and Vocabulary. Geneva: ISO, 2008.

FARIA, Leandro. **Scrum of Scrums: Running Agile on Large Projects.** 2013. Disponível em: <https://www.scrumalliance.org/community/articles/2013/june/scrum-of-scrums-running-agile-on-large-projects>. Acesso em: 28 dez. 2015.

GÄRTNER, Markus. **ATDD by Example: A Practical Guide to Acceptance Test-Driven Development.** Westford: Addison-Wesley Professional, 2012.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social.** 6. ed. São Paulo: Atlas, 2008.

GLASS, Robert L.. Defining Quality Intuitively. **IEEE Software**. [S. l.], p. 103-107. maio 1998.

HAMMOND, Susan; UMPHRESS, David. Test driven development: the state of the practice. In: ACM SOUTHEAST REGIONAL CONFERENCE, 50., 2012, Tuscaloosa. **Proceedings...** New York: ACM, 2012. p. 158 - 163.

HENDRICKSON, Elisabeth. **Driving Development with Tests: ATDD and TDD.** [2008]. Disponível em: <http://testobsessed.com/wp-content/uploads/2011/04/atddexample.pdf>. Acesso em: 13 jan. 2016.

KOSKELA, Lasse. **Test Driven: TDD and Acceptance TDD for Java Developers.** Greenwich: Manning Publications, 2008.

LASKOSKI, Felipe Cys; RADAELLI, Lucas Falcão. **Desenvolvimento de Software Modelo Incremental.** Paraná, [2012]. 11 slides, color. Disponível em: <http://www.inf.ufpr.br/lmperes/ciclos_vida/ModeloIncremental.pdf>. Acesso em: 13 jan. 2016.

LINDSTROM, Lowell; JEFFRIES, Ron. Extreme Programming and Agile Software Development Methodologies. **Information Systems Management.** [S. l.], p. 41-52. 2004.

LIVERMORE, Jeffrey A.. Factors that impact implementing an agile software development methodology. In: SOUTHEASTCON., 2007, Richmond. **Proceedings... .** [S. l.]: IEEE, 2007. p. 82 - 86.

MCCALL, Jim A.; RICHARDS, Paul K.; WALTERS, Gene F.. **Factors in Software Quality.** New York: , 1977.

MCLAUGHLIN, Mike. **Agile Methodologies.** [2015]. Disponível em: <https://www.versionone.com/agile-101/agile-methodologies/>. Acesso em: 10 jan. 2016.

NORTH, Dan. Behavior Modification: The Evolution of Behavior-Driven Development. **Better Software,** [S. l.], v. 8, p.26-31, mar. 2006. Trimestral. Disponível em: <http://www.stickyminds.com/sites/default/files/magazine/file/2012/XDD10836filelistfilename1_0.pdf>. Acesso em: 03 jan. 2016.

ORAM, Andy; WILSON, Greg. **Making Software:** What Really Works, and Why We Believe It. Sebastopol: O'reilly Media, 2011.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma abordagem Profissional. 7. ed. Porto Alegre: AMGH, 2011.

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK®).** 5. ed. Newtown Square: Project Management Institute, 2013.

PUTNAM-MAJARIAN, Taylor; PUTNAM, Doug. **The Most Common Reasons Why Software Projects Fail.** 2015. Disponível em: <http://www.infoq.com/articles/software-failure-reasons>. Acesso em: 02 jan. 2016.

REBELO, Paulo. **Acceptance Test-Driven Development (ATDD), passo a passo.** 2014. Disponível em: <http://www.infoq.com/br/articles/atdd-passo-a-passo>. Acesso em: 12 jan. 2016.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software.** Rio de Janeiro: Alta Books, 2013. 304 p.

SAUVÉ, Jacques Philippe; ABATH NETO, Osório Lopes. Teaching Software Development with ATDD and EasyAccept. In: SIGCSE TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, 39., 2008, New York. **Proceedings... .** New York: IEEE, 2008. p. 542 - 546.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide.** 2013. Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em: 10 jan. 2016.

SERTA, Victor. **BDD em aplicações Web - Revista Java Magazine 98.** 2011. Disponível em: <http://www.devmedia.com.br/bdd-em-aplicacoes-web-revista-java-magazine-98/23071>. Acesso em: 09 jan. 2016.

SOLÍS, Carlos; WANG, Xiaofeng. A Study of the Characteristics of Behaviour Driven Development. In: EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS (SEAA), 37., 2011, Oulu. [S. l.]: IEEE, 2011. p. 383 - 387.

SOMMERVILLE, Ian. **Engenharia de Software.** 8. ed. São Paulo: Pearson - Addison Wesley, 2007.

VIEIRA, Juraci; PUFAL, Nicholas. **Três falácias sobre BDD.** 2013. Disponível em: <https://www.thoughtworks.com/pt/insights/blog/3-misconceptions-about-bdd>. Acesso em: 07 jan. 2016.

WELLS, Don. **Extreme Programming: A gentle introduction.** 2013. Disponível em: <http://www.extremeprogramming.org/>. Acesso em: 10 jan. 2016.

APÊNDICE A - Questionário utilizado para coletar os dados dessa pesquisa

Pesquisa sobre TDD e Extensões no contexto brasileiro

Prezado(a)s,

Meu nome é Karla Silva, sou aluna da graduação em Sistemas de Informação da Universidade Federal de Pernambuco e estou desenvolvendo o meu trabalho de conclusão de curso (TCC) sob orientação da professora Dra. Carla Silva.

Essa pesquisa visa entender melhor o contexto do uso do BDD, DbC, TDD e outras extensões no mercado brasileiro. Se você já trabalhou com alguma dessas técnicas, você está convidado a responder essa pesquisa.

De acordo com testes realizados previamente, o tempo estimado para responder esta pesquisa é entre 2 e 3 minutos.

Se você tiver qualquer dúvida em relação à pesquisa, por favor, entre em contato com a pesquisadora Karla Silva, por meio do e-mail: kmbs@cin.ufpe.br

*Obrigatório

Termo de Consentimento Livre e Esclarecido (TCLE) *
Sua participação não é obrigatória. A qualquer momento, você poderá desistir de participar e retirar seu consentimento. Sua recusa, desistência ou retirada de consentimento não acarretará prejuízo. Os dados obtidos por meio desta pesquisa serão confidenciais e não serão divulgados em nível individual, visando assegurar o sigilo de sua participação. A pesquisadora responsável se comprometeu a tornar públicos nos meios acadêmicos e científicos os resultados obtidos de forma consolidada sem qualquer identificação de indivíduos ou instituições participantes. Caso você concorde em participar desta pesquisa, selecione o checkbox de “Concordo” e clique no botão “Próxima”.

☐ Concordo

Próxima

Seção 2

Pesquisa :: TDD e Extensões

Você já utilizou alguma das técnicas abaixo? *

☐ Acceptance Test-Driven Development (ATDD)
☐ Behavior-Driven Development (BDD)
☐ Test Driven Development (TDD)

- [ ] Design-by-Contract (DbC)
- [ ] Outro: _____

Você trabalhava em qual estado quando utilizou a(s) técnica(s) citada(s) na pergunta 1? *
Exemplo de Resposta: TDD - São Paulo e BDD - Pernambuco

Sua Resposta

Onde você trabalhava quando utilizou a(s) técnica(s) citada(s) na pergunta 1? *
Exemplo de Resposta: TDD - Accenture e BDD - Thoughtworks Brasil

Sua Resposta

Qual o papel que você desempenhava no projeto quando utilizou a(s) técnica(s) citada(s) na pergunta 1 ? *
Exemplo de Resposta: TDD - Desenvolvedor e BDD - Engenheiro de Requisitos

Sua Resposta

Quantas pessoas (mais ou menos) trabalhavam no projeto quando você utilizou a(s) técnica(s) citada(s) na pergunta 1? *
Exemplo de Reposta: TDD - Entre 10 e 20 pessoas e BDD - Entre 20 e 30 pessoas

Sua Resposta

VOLTAR PRÓXIMA

Seção 3

Experiência sobre a adoção do TDD e outras extensões

Seu time encontrou dificuldades na adoção da(s) prática(s) escolhida(s)? *

- [ ] Sim
- [ ] Não
- [ ] Talvez

Para cada prática usada, quais foram as principais dificuldades que o seu time encontrou no processo de adoção?

Sua Resposta

Foram identificados benefícios após a adoção? *

- [ ] Sim
- [ ] Não
- [ ] Talvez

Para cada prática usada, quais foram os benefícios a curto e longo prazo identificados?

Sua Resposta

Foram identificadas mudanças no projeto/time/empresa após a adoção? *

- [ ] Sim
- [ ] Não
- [ ] Talvez

Para cada prática usada, quais foram as principais mudanças identificadas?

Sua Resposta

Foram usadas ferramentas durante o processo? *

- [ ] Sim
- [ ] Não

Para cada prática usada, quais foram as ferramentas utilizadas durante e após a adoção?

Sua Resposta

Você deseja receber os resultados dessa pesquisa?
Se sim, só basta escrever um e-mail para contato no campo abaixo. Se não, pode clicar no botão "Enviar" para finalizar a pesquisa.

Sua Resposta

VOLTAR ENVIAR