Pós-Graduação em Ciência da Computação

“Em Direção a um Ambiente de Desenvolvimento de Software Orientado por Comportamento”

Por

ALVARO MAGNUM BARBOSA NETO

Dissertação de Mestrado Profissional

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE

2015

ALVARO MAGNUM BARBOSA NETO

EM DIREÇÃO A UM AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE ORIENTADO POR COMPORTAMENTO

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre Profissional em Ciência da Computação.

ORIENTADOR: Prof. Dr. Vinicius Cardoso Garcia

RECIFE

2015

Catalogação na fonte
Bibliotecária Joana D'Arc Leão Salvador CRB4-532

B238e Barbosa Neto, Alvaro Magnum.
Em direção a um ambiente de desenvolvimento de software orientado por comportamento / Alvaro Magnum Barbosa Neto. – Recife: O Autor, 2015.
106 f.: fig., tab.

Orientador: Vinicius Cardoso Garcia.
Dissertação (Mestrado Profissional) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.
Inclui referências e apêndices

1. Engenharia de software. 2. Software - testes. I. Garcia, Vinicius Cardoso (Orientador). II. Titulo.

005.1 CDD (22. ed.) UFPE-MEI 2015-108

Dissertação de Mestrado Profissional apresentada por Álvaro Magnum Barbosa Neto à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título, “Em Direção a um Ambiente de Desenvolvimento de Software Orientado por Comportamento”, orientada pelo Professor Vinícius Cardoso Garcia e aprovada pela Banca Examinadora formada pelos professores:

_____

Profª. Carla Taciana Lima Lourenço Silva Schuenemann

Centro de Informática / UFPE

_____

Prof. Leandro Marques do Nascimento

Universidade Federal Rural de Pernambuco

_____

Prof. Vinícius Cardoso Garcia

Centro de Informática / UFPE

Visto e permitida a impressão.

Recife, 22 de maio de 2015.

_____

Profª. EDNA NATIVIDADE DA SILVA BARROS

Coordenadora da Pós-Graduação em Ciência da Computação do

Centro de Informática da Universidade Federal de Pernambuco.

Dedico este trabalho a toda minha família, principalmente a minha esposa que está sempre do meu lado e a quem amo tanto. Dedico também a meus pais e a minha irmã.

AMO VOCÊS.

Agradecimentos

Inicialmente, agradeço ao único que é digno de toda honra e toda glória: Deus. Agradeço o dom da vida, a família que Ele me deu e, principalmente, Sua graça e misericórdia que abundam a vida deste grande pecador.

Agradeço a minha esposa, Roberta Arruda, que me incentivou e cobrou perseverança ao longo de todo trabalho. Sem seu amor, carinho e cuidados muitas de minhas conquistas não teriam o mesmo sabor que têm hoje. Agradeço sua paciência e tolerância todas as vezes em que tive que me dedicar a pesquisa e não a ela. Agradeço sua ajuda, pois, ainda que da área do Direito, e não de Tecnologia, procurou e indicou pessoas conhecidas de TI para avaliarem meu trabalho. Te amo demais. Juntos, somos UM.

Aos meus pais, Alvaro Barbosa e Marta Sueli. Muitos de meus princípios e educação foram enraizados por eles. Seus exemplos e condutas influenciaram a minha vida. A maior parte de minha vida passei ao lado deles, foram momentos de muitas alegrias e aprendizados. Meus pais nunca deixaram de me incentivar a crescer tanto acadêmica, quanto profissionalmente. Por estes motivos e muitos outros, a eles dou honra. Amo vocês.

A minha irmã, Marta Monike, por quem tenho grande amor e carinho. Sua atenção e seu jeito meigo de me tratar me dão forças para fazer muitas coisas. Ela me incentivou bastante com suas especializações e sua busca por crescimento no meio acadêmico. Amo você.

Ao meu orientador, o Dr. Vinicius Cardoso, que sempre falou claramente o que "prestava" e o que "não prestava" em minhas pesquisas e por me ensinar que, independentemente do problema, temos que contornar e ir em frente. Todos passam por dificuldades e somos responsáveis por enfrenta-los e vencê-los. Sua objetividade, clareza e simplicidade em nortear o caminho, se mostraram muito desafiadores e estimulantes: algo, mais ou menos, como "O caminho é esse, o resto depende de você e não de mim".

Aos professores do curso que, pela experiência de fazerem parte de uma das universidades federais mais importantes do país, e com grande destaque no cenário nacional de tecnologia, nos orientaram e nos incentivaram com paixão e afinco. Souberam nos mostrar como eliminar o descartável e focar no que é importante e, principalmente, que nem sempre o caminho será fácil.

Aos amigos e colegas da turma do mestrado profissional que se uniram durante todo o curso, estudaram e trocaram ideias. Em especial agradeço a "turma" de João Pessoa: Ernandes e Tales. Montamos uma equipe "arrasadora" do início ao fim. Estudamos, trabalhamos, sofremos e nos alegramos juntos. Infelizmente por duas notas B, não ficamos com 100% de nota A. Quem sabe em um doutorado.

Aos companheiros de profissão da Paraíba Previdência que adotaram a ferramenta (que foi um dos frutos desse trabalho) dentro do ambiente corporativo e procuraram utilizar e se beneficiar dela, contribuindo com dados estatísticos e ideias.

"Eu sou o Alfa e o Ômega, o Primeiro e o Último, o Princípio e o Fim. Felizes as pessoas que lavam as suas roupas, pois assim terão o direito de comer a fruta da árvore da vida e de entrar na cidade pelos seus portões!"

Apocalipse 22:13-14

Resumo

Criado por Dan North, o BDD (Behavior Driven Development) é uma técnica de desenvolvimento ágil de software baseada no TDD (Test Driven Development) e que foca no teste de software orientado por comportamentos, isto é, concentra-se nas razões pelo qual o software é criado e nos requisitos de comportamento do negócio. A utilização da técnica traz uma série de benefícios para projetos de desenvolvimento de software, contudo, ela não tem uma aceitação tão grande no mercado e é, muitas vezes, preterida em relação ao TDD. Esse trabalho faz uma análise dessa situação e também propõe um ambiente que visa facilitar a adoção do BDD através da análise dos seguintes questionamentos: quais características devem fazer parte de uma ferramenta para que ela facilite e dinamize a utilização do BDD no contexto de um projeto de desenvolvimento de software? Como permitir o uso da mesma por um cliente leigo em testes, e, ao mesmo tempo, agregar valor para o gerente do projeto, os testadores e os desenvolvedores de software? Como o cliente poderia acompanhar em tempo real se o que ele espera obter está, de fato, sendo construído? Como medir o impacto da ferramenta? Através de análises e resultados obtidos em mais de 12 anos de experiência profissional no setor de tecnologia de instituições públicas e privadas, além de pesquisas na literatura, entrevistas com profissionais de TI e avaliações de ferramentas BDD no mercado, foi concebido um plugin: o BDD Plugin for Mantis (BDDPM), uma ferramenta cujo objetivo é facilitar a adoção do BDD em projetos de desenvolvimento de software. Para avaliar o plugin quanto ao cumprimento dos objetivos, foi utilizada uma técnica denominada GQM (Goal/Question/Metric), que permite, através de objetivos bem estabelecidos, planejar e mensurar métricas de avaliação. O BDDPM foi avaliado com sucesso dentro de um ambiente de produção real, uma autarquia do Governo do Estado da Paraíba: a Paraíba Previdência. Este trabalho descreve, em detalhes, todo o ciclo de vida do projeto, desde sua concepção, passando por sua criação, tecnologias utilizadas, recursos incluídos, etc.

Palavras-chave: Plugin. BDD. Mantis. Testes. Software. Projeto. Desenvolvimento. GQM. Análise. Medição. Adoção do BDD. Acompanhamento de Testes. Escrita de Testes.

Abstract

Created by Dan North, BDD (Behavior Driven Development) is a software agile development technique based on TDD (Test Driven Development). The BDD focuses on software testing oriented by behaviors, that is, it focuses on the reasons why a software is created and its business behavior. The use of the technique brings a number of benefits for software development projects; however, BDD does not have such a great market as the TDD: the first choice of the majority. This work brings an analysis of this situation and also proposes an environment to facilitate the adoption of BDD by examining the following questions: what characteristics should be part of a tool so that it facilitate and streamline the use of BDD in a context of project software development? How can it be used by an unexperienced client, and, at the same time, add value to project managers, testers and developers? How the customer could follow, in real time, if what he expects to, is really being built? How to measure the impact of the tool? Through analysis and results obtained from over 12 years of professional experience in the technology sector of public and private institutions, as well as research in the literature, interviews with IT professionals and reviews of BDD tools on the market, a plugin was developed: the BDD Plugin for Mantis (BDDPM), a tool which aims to facilitate the adoption of BDD in software development projects. To assess the plugin in meeting the goals, a technique called GQM (Goal / Question / Metric) was used; it allows, through well-established objectives, plan and measure evaluation metrics. The BDDPM was successfully evaluated in a real production environment, a company called Paraíba Previdência. This paper describes in detail the entire life cycle of the project: from its conception, through its creation, the technologies used, features included, etc.

Keywords: Plugin. BDD. Mantis. Tests. Software. Project. Development. GQM. Analysis. Measurement. BDD Adoption. Test Tracking. Test Writing.

Lista de Abreviações

ABNT	Associação Brasileira de Normas Técnicas
ACM	Association for Computing Machinery
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASF	Apache Software Foundation
BDD	Behavior Driven Development
BDDPM	Behavior Driven Development Plugin for Mantis
BSD	Berkeley Software Distribution
BT	Bug Tracker
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CESAR	Centro de Estudos e Sistemas Avançados do Recife
CMMI	Capability Maturity Model Integration
COC	Convention over Configuration
CSS	Cascading Style Sheets
DOM	Document Object Model
DR	Doutor
DRY	Don't Repeat Yourself
DSL	Domain-Specific Language
FSF	Free Software Foundation
GNOME	GNU Network Object Model Environment
GPL	General Public License
GQM	Goal-Question-Metric
GT	Grounded Theory
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JIT	Just In Time
JR	Júnior
JS	JavaScript
JUG	Java User Group
MIT	Massachusetts Institute of Technology
MPROF	Mestrado Profissional
MVC	Model-View-Controller
NASA	National Aeronautics and Space Administration
ORM	Object Relational Mapping
PBPREV	Paraíba Previdência
PHP	Hypertext Preprocessor
PL	Pleno
PMBOK	Project Management Body of Knowledge
QA	Quality Assurance
RBAC	Role-Based Access Control
RIA	Rich Internet Application
RPC	Remote Procedure Call

RSS	Rich Site Summary
SCM	Source Code Management
SEO	Search Engine Optimization
SGBD	Sistema de Gerenciamento de Banco de Dados
SLA	Service Level Agreement
SR	Sênior
SSAC	SISPROTO - Solicitações de Alteração e Correção
SVN	Subversion
TDD	Test Driven Development
TI	Tecnologia da Informação
UFCG	Universidade Federal de Campina Grande
UFPE	Universidade Federal de Pernambuco
UI	User Interface
URL	Uniform Resource Locator
VCS	Version Control System
VS	Visual Studio
WWW	World Wide Web
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language

Lista de Tabelas

Lista de Figuras

Sumário

Capítulo 1 – Introdução

> "Eu tive um problema. Enquanto estive usando e ensinando práticas ágeis como o desenvolvimento orientado a testes (TDD[1]) em projetos em diferentes ambientes, sempre me deparei com as mesmas confusões e mal-entendidos: programadores queriam saber por onde começar, o que testar e o que não testar, quanto testar, como denominar os testes e como entender por que um teste falha."
>
> [DAN NORTH, 2006]

Este capítulo faz uma apresentação geral deste trabalho e do problema a ser tratado justificando-o e elucidando sua proposta de solução. Também descreve as técnicas escolhidas, tanto para o desenvolvimento da ferramenta fruto do esforço, bem como para avaliação dos resultados.

O capítulo inicia-se na seção 1.1 com um breve sumário do cenário de testes de software com BDD e TDD. Em seguida, na seção 1.2, o problema foco deste trabalho é explicitado para, na sequência (seção 1.3), mostrar uma visão geral da proposta para "tratamento" do problema. Por fim, a Seção 1.4 descreve a metodologia aplicada na consecução do trabalho e a 1.5 discorre sobre a organização desta dissertação de maneira a facilitar sua leitura e descrever, resumidamente, o que pode ser encontrado em cada capítulo.

1.1. Teste de Software: TDD e BDD na Prática

O objetivo do teste de software é investigar o software com o objetivo de avaliar sua qualidade dentro de um domínio/contexto. Dentro do processo encontra-se a atividade de identificação de falhas e defeitos [Sommerville, 2011]. Estima-se que, atualmente, 50% do tempo e do custo de projetos de software são dispendidos em teste de software, um percentual considerável. Entretanto, mesmo considerando o impacto dos testes, ou falta deles, na qualidade do software, observa-se que há um gargalo entre a teoria que é vista nas universidades e o que é praticado no mercado nas fábricas de software. Na prática, a teoria tem que ser adaptada. Estas são

[1] Test Driven Development (TDD) ou em português Desenvolvimento orientado a testes é uma técnica de desenvolvimento de software que se baseia em um ciclo curto de repetições: primeiramente o desenvolvedor escreve um caso de teste automatizado que define uma funcionalidade ou melhoria desejada em uma função. Em seguida é produzido um código que possa ser validado pelo teste para, posteriormente, o código ser refatorado para uma versão sob padrões aceitáveis.

afirmações de profissionais com mais de 15 anos de mercado na área de educação em Teste de Software na obra "Software Testing and Quality Assurance" de Kshirasagar Naik e Priyadarshi Tripathy. Experiência semelhante pode ser encontrada em relato disponível no ANEXO IV deste documento. Ainda em consonância com estas informações, existem dois casos emblemáticos de grandes empresas na área de tecnologia (Google e Microsoft), que corroboram com esta avaliação.

O artigo “How Microsoft Builds Software” (http://goo.gl/kvmLWZ), publicado na Magazine Communications of the ACM por Michael A. Cusumano (49 publicações, 313 citações e média de 1.974 downloads por publicação) e Richard W. Selby (36 publicações, 626 citações e média de 1.844 downloads por publicação) fala sobre uma das metodologias de desenvolvimento de software utilizada pela Microsoft. Basicamente, a estratégia pela empresa é manter o contato constante entre desenvolvedores e testadores para facilitar a troca de informações e garantir que a criação dos testes seja a mais abrangente possível. Essa técnica é intitulada de "Aproximação Baseada em Sincronização e Estabilização". Ela também adota uma outra técnica, denominada “Engenharia Conectada ao Cliente”. Na prática, a Microsoft utiliza metodologias próprias para desenvolvimento, teste e qualidade de software. Estas metodologias se baseiam nas melhores práticas do mercado. Em 2012 a Microsoft realizou uma série de visitas às Universidades Federais do Brasil (https://goo.gl/bd2cca) e reforçou a ideia de metodologias próprias (http://goo.gl/hTFyb).

Seguindo o mesmo caminho, a Google fez uma apresentação para o DallasJUG (http://goo.gl/IRnemv), um grupo de usuários Java, onde falou a respeito de como utiliza as metodologias ágeis de desenvolvimento de software. No evento a empresa classificou sua metodologia como uma "metodologia própria" e afirmou utilizar as melhores práticas encontradas no mercado para desenvolvimento orientado por testes, contato constante com o cliente, iterações curtas, definição de processos claros, etc. Justificou a necessidade de uma metodologia própria com a resposta de que, para ela, as práticas tradicionais são muito eficientes para projetos de pequeno e médio porte, porém, para projetos de grande porte, é necessário fazer algumas adaptações para manter a eficácia e a eficiência.

De acordo com Dan North (http://goo.gl/HSzCm), geralmente as empresas adotam testes de unidade com a abordagem do TDD[2] como requisito básico e primordial para testes. Cada desenvolvedor “escreve” seu código juntamente com uma bateria mínima de testes de unidade. Depois o software passa por uma bateria de testes exaustivos junto aos testadores com o objetivo de encontrar defeitos. Nesse sentido, há muito esforço para evitar que um único erro passe despercebido, porém sem preocupação se a funcionalidade “construída” satisfazia a necessidade cliente. Uma possível explicação para as coisas serem assim pode ser o fato de que, muitas vezes, manter o contato constante com o cliente é custoso: ele pode estar localizado geograficamente em lugar divergente do local da produção do software e/ou ele pode não ter tempo para este contato constante e/ou ocorrem dificuldades na tradução da

[2] Test Driven Development (TDD - Desenvolvimento orientado por teste) é uma técnica de desenvolvimento de software que se baseia em um ciclo curto de repetições: primeiro cria-se o teste, depois o código, depois refatora-se o código.

linguagem de negócio de alto nível deste cliente (requisitos de usuário) para uma linguagem técnica (requisitos de sistema). Sob esta perspectiva, é muito fácil descrever e especificar funções técnicas, por isso o TDD é tão utilizado. Por outro lado, quando se fala em BDD[3], ou seja, o comportamento do software, não há tantos esforços em sua prática: o BDD gera, pelo menos, 75% a menos de resultados em sites de pesquisas comerciais e acadêmicos, por exemplo.

Quando Dan North concebeu o Behavior Driven Development, seu objetivo era buscar uma maior aproximação com o cliente do produto de software durante o desenvolvimento do mesmo, pois o cliente é um dos melhores stakeholders[4] para descrever o que ele mesmo espera do produto que está adquirindo. Infelizmente, conforme citado anteriormente, a prática do mercado mostrou que era muito complicado trabalhar diretamente com o cliente. Surge, então, a figura do Analista de Requisitos, que é o responsável por fazer a ponte ente o que o cliente deseja, que é descrito em alto nível, e a sua correspondente especificação técnica, voltada para desenvolvedores e testadores de software. Com isso, clientes de software se distanciaram mais dos projetos de desenvolvimento de software e, consequentemente do teste de software, e os analistas de requisitos ganharam mais responsabilidades.

O Cucumber[5] foi um grande passo na tentativa de reaproximar o cliente da fase de testes de seu produto. Ao utilizar uma linguagem específica de domínio e "Business Readable", o Gherkin, um novo chamado ao maior engajamento dos clientes em testes foi feito. Este tipo de linguagem foca na possibilidade de leitura e no entendimento por parte do cliente. Com essa linguagem eles podem descrever o comportamento esperado do software.

Apesar dos benefícios, o BDD não se aproxima do TDD em termos de aceitação. Quais os motivos que levaram a este fato? Por que as empresas priorizam funcionalidades em detrimento do comportamento? Mas, principalmente, como seria possível possibilitar uma maior adoção do BDD? Estes são alguns dos questionamentos discutidos ao longo deste trabalho.

1.2. Explicitando o Problema

Com base no cenário descrito anteriormente, e além dos questionamentos já apresentados, as seguintes perguntas também foram levantadas:

[3] Behavior Driven Development (BDD - Desenvolvimento orientado por comportamento) é uma técnica de desenvolvimento de software oriunda do TDD e que segue os mesmos princípios desse, entretanto foca no comportamento do software esperado pelo cliente.

[4] Stakeholder (em português, parte interessada ou interveniente), é um termo usado em diversas áreas como gestão de projetos, administração, engenharia de software, etc., e se refere a toda e qualquer parte interessada e/ou afetada em um determinado âmbito, contexto, projeto, empreendimento. Pode englobar tanto participantes externos, quanto internos.

[5] O Cucumber é uma ferramenta de software utilizada por programadores/desenvolvedores para testar outros softwares. O Cucumber foi escrito em Ruby e executa testes de aceitação BDD em uma linguagem de alto nível e semiformal (Gherkin). O Cucumber permite o teste de aplicações criadas em diferentes plataformas (Java, Ruby, PHP, .NET, etc.).

1. **Quais características devem fazer parte de uma ferramenta para que ela facilite e dinamize a utilização do BDD no contexto de um projeto de desenvolvimento de software?**
2. **Como permitir o uso da mesma pelo cliente do produto de software, facilitando sua aproximação dos testes, e, ao mesmo tempo, agregar valor para o gerente do projeto, os testadores e os desenvolvedores de software?**
3. **Que características devem fazer parte de uma solução para que ela facilite a adoção do BDD?**
4. **Como medir o impacto e sucesso da ferramenta proposta?**

Foi observado que o BDD é, muitas vezes, ignorado em detrimento do TDD e que o contato com o cliente e o acompanhamento (por este mesmo cliente) dos testes do projeto em tempo real são falhos e podem ser custosos para uma empresa. As organizações esperam entregar um sistema livre de defeitos, mas não priorizam, com o mesmo afinco, a questão comportamental do software. Dessa maneira, constroem-se produtos com funcionamento LIVRE de falhas e comportamento CHEIO de falhas. Este fato, aliado às perguntas, convergem para um questionamento mais amplo, "**Como ir em direção a um ambiente de desenvolvimento de software orientado por comportamento?",** que dá nome a este trabalho. A figura a seguir retrata o problema:

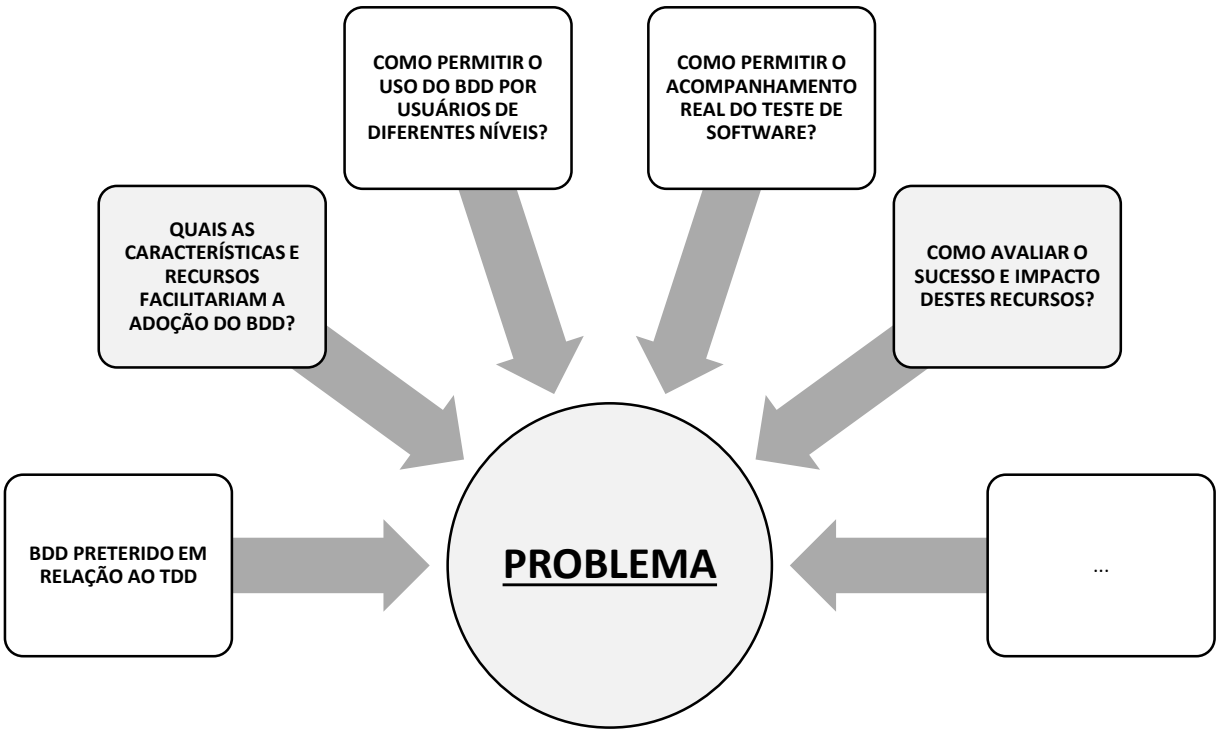

Figura 1 – Questionamentos e fatos que culminam no problema "Como ir em Direção a um Ambiente de Desenvolvimento de Software Orientado por Comportamento?"

1.3. Visão Geral da Proposta

A solução para o problema deveria ser “algo” que respondesse, ou ajudasse a responder, as perguntas enumeradas anteriormente e que vão ao encontro do ambiente desejado: um ambiente orientado por comportamento. Foram estabelecidos um conjunto de recursos que auxiliam e automatizam boa parte do processo com o BDD (participação do cliente, escrita de testes, acompanhamento de testes, geração automática de códigos, etc.) com o objetivo de facilitar sua adoção. Estes recursos foram encapsulados em uma ferramenta de software: O BDDPM, um dos frutos deste trabalho.

Ainda como parte da proposta, os recursos incorporados na ferramenta foram rigorosa e sistematicamente avaliados a fim de se entender seus resultados e impactos como solução do problema. Esta avaliação foi feita com o GQM[6] dentro de um ambiente de produção real. As características do GQM, detalhado no Capítulo 5, facilitam avaliação de projetos desse tipo, onde, a princípio, não se sabe o que deve ser medido em relação ao produto de software.

As figuras 2 e 3 retratam a proposta:

Figura 2 – Visão Geral da Proposta: Recursos → Adoção BDD → Avaliação

[6] GQM é uma abordagem para o estabelecimento de um sistema de medição (levantamento de métricas de avaliação), normalmente utilizada na área de software.

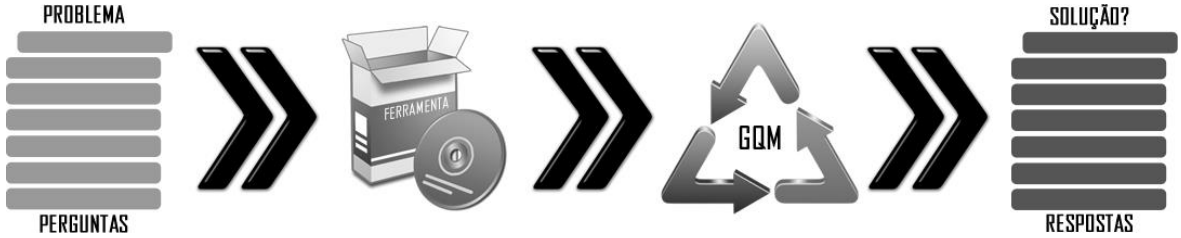

Figura 3 – Visão Geral da Proposta: Ferramenta → GQM → Impactos

Para aplicar a proposta dentro de limites razoáveis de pesquisa, o escopo do problema foi estreitado, minimizando a quantidade de variáveis e cenários. O seguinte escopo foi definido:

1. **Seria criado um plugin para uma plataforma de gestão/acompanhamento de projetos de software já estabelecida no mercado. A plataforma escolhida foi o "Mantis Bug Tracker"**
2. **O tipo dos projetos a serem testados com a ferramenta deveriam ser "Projetos de Desenvolvimento de Software". A linguagem dos projetos deveria ser C#.NET (Microsoft).**
3. **A ferramenta seria testada em um único ambiente e conjecturas sobre sua utilização em maior escala seriam estabelecidas.**

O Mantis é um sistema de gestão WEB para acompanhamento de problemas e defeitos de (geralmente) software e projetos de software. É uma ferramenta "Open Source"[7] construída com a linguagem de programação PHP e que roda em Windows, Linux ou Mac OS X. Por este motivo, a ferramenta, que recebeu o nome de "BDD Plugin for Mantis – BDDPM", foi construída na mesma linguagem.

O Mantis foi a plataforma escolhida pela familiaridade da equipe de desenvolvimento com a linguagem PHP. Além disso, na página de listagem de plugins disponíveis para o sistema no site oficial da ferramenta, não existem plugins que ofereçam suporte a BDD. Apesar da escolha, muitas das funcionalidades do BDDPM estão no Front-End[8] da aplicação com Java Script para facilitar portabilidade futura para outras plataformas como o Jira, Mantis, RedMine, Trac, BugZilla, etc.

A linguagem de programação dos projetos (C#.NET) também foi escolhida pela familiaridade da equipe de desenvolvimento com a mesma. De acordo com o "Ranking da TIOBE"[9], PHP e C# possuem, juntas, 7.074% do mercado; sendo, respectivamente, a 5ª e a 6ª linguagens mais utilizadas.

[7] De acordo com a OSI (Open Source Initiative - http://opensource.org) um software é dito Open Source (Código Aberto) quando pode ser livremente utilizado, modificado e compartilhado, tanto a versão original quanto a modificada. A OSI é a atual mantenedora do Open Source.

[8] Em ciência da computação, front-end e back-end são termos generalizados que se referem às etapas inicial e final de um processo. O front-end é responsável por coletar a entrada em várias formas do usuário e processá-la para adequá-la a uma especificação em que o back-end possa utilizar. O front-end é uma espécie de interface entre o usuário e o back-end.

[9] O Ranking da TIOBE (http://tiobe.com/index.php/content/paperinfo/tpci/index.html) é um renomado indicador da popularidade das linguagens de programação do mercado. O índice é atualizado uma vez por mês. Os resultados são baseados no número de usuários/desenvolvedores ao redor do globo, cursos e fabricantes de software, além de número de resultados nos principais buscadores da WEB.

Posição	Linguagem de Programação	Índice
1	C	17.588%
2	Java	14.959%
3	Objective-C	9.130%
4	C++	6.104%
5	C#	4.328%
6	PHP	2.746%
7	Java Script	2.433%
8	Python	2.287%
9	Visual	2.235%
10	Perl	1.826%

Tabela 1 – Ranking (TOP 10) de Linguagens de Programação da TIOBE (Dezembro/2014)

1.4. A Metodologia de Pesquisa Aplicada

O principal objetivo é ir em direção ao um ambiente de desenvolvimento de software orientado por comportamento. Os recursos BDD com objetivo de facilitar a adoção da técnica foi feita após Pesquisa Bibliográfica descrita neste documento, bem como pesquisa com 56 profissionais da área de TI, sendo 47 desenvolvedores / testadores, 06 gestores de projetos de TI e 03 clientes de produtos de software. O questionário pode ser visto no Anexo 1 deste documento. Este conjunto de recursos / funcionalidades foram agregados a uma ferramenta, o BDDPM (Behavior Driven Development for Mantis), entretanto, apenas a criação de uma ferramenta não seria suficiente para avaliação dos resultados. Este produto teria que ser implantado em um projeto real e técnicas de avaliação consolidadas teriam que ser empregadas para avaliação de resultados e estabelecimento de hipóteses sólidas, inclusive com conjecturas para utilização da ferramenta em larga escala. Como isso, seria possível entender os níveis de impacto da ferramenta como solução para o problema apresentado.

Antes do início do desenvolvimento da ferramenta, foi feita uma avaliação de mercado para se entender como o TDD e o BDD estavam sendo utilizadas dentro das empresas no contexto da Gestão de Projetos de Desenvolvimento de Software. O TDD foi incluído na pesquisa, pois o BDD tem suas origens nesta técnica; além disso, este comparativo evidenciou o quão incipiente se encontra a utilização do Behavior Driven Development em detrimento do TDD, além da escassez de ferramentas (plugins e extensões) para utilização do BDD em projetos de software, o que seria mais uma justificativa para a criação do BDDPM.

Após a criação e implantação do BDDPM em um ambiente de produção real, a saber, a Paraíba Previdência – PBprev[10], que autorizou a divulgação de seu nome e resultados da avaliação (Anexo 3) que foi obtida através da aplicação da técnica GQM após a experimentação do plugin em um projeto de desenvolvimento de software real. O Capítulo 5 descreve a técnica e detalha como ela foi utilizada para avaliação da ferramenta proposta.

[10] http://www.pbprev.pb.gov.br

Este é o contexto de criação e avaliação do BDDPM:

1. **Equipe de desenvolvimento do BDDPM:**
   - 01 Analista de Sistemas: Alvaro Magnum Barbosa Neto (ambn@cin.ufpe.br), 30 anos, Gestor de TI, 12 anos de experiência.
   - 01 Supervisor/Coordenador: Dr. Vinicius Cardoso Garcia (vcg@cin.ufpe.br), 36 anos, Professor Adjunto - UFPE, 18 anos de experiência.
2. **Tempo de desenvolvimento do BDDPM:** 90 dias
3. **Empresa onde o BDDPM foi implantado:** PBprev
4. **Projeto no qual o BDDPM foi utilizado**: SISPROTO – Solicitações de Alteração e Correção (SISPROTO – SAC → SSAC)[11]
5. **Equipe de desenvolvimento do SISPROTO – SAC:**
   - 01 Analista de Sistemas: Daniel de Oliveira Fernandes (mdof@pbprev.pb.gov.br), 24 anos, Coordenador de TI, 03 anos de experiência.
   - 02 Estagiários[12]: Marcos Fernando Carneiro (mmfc@pbprev.pb.gov.br) – 26 anos, Estagiário, 06 meses de experiência; e João Paulo Cordeiro (mjpc@pbprev.pb.gov.br) – 27 anos, Estagiário, 06 meses de experiência.
6. **Tempo de desenvolvimento do SISPROTO – SAC:** 45 dias[13]
7. **Equipe de Avaliação do BDDPM:** Mesma do Item 1
8. **Tempo de Avaliação do BDDPM:** 30 dias

1.5. Estrutura e Resumo da Dissertação

Esta dissertação está dividida em 04 partes: a primeira parte versa sobre o contexto, a situação do BDD e ferramentas relacionadas no mercado; e os conceitos e teorias necessárias para o entendimento do trabalho. A segunda parte descreve o BDDPM em maiores detalhes, incluindo todo o processo de criação do plugin, componentes e tecnologias utilizadas, licenças envolvidas, funcionalidades, etc. A terceira envolve a avaliação do projeto, bem como conclusões e comentários sobre os resultados após a utilização da ferramenta dentro de um ambiente de produção real: a PBprev, uma autarquia do governo do Estado da Paraíba que tem como objetivo gerir o regime próprio de previdência dos servidores públicos estaduais. Por fim, na última parte (PARTE IV), encontram-se as referências bibliográficas e os anexos.

Todos os capítulos deste documento contam com um resumo como introdução.

Descrição sumária do conteúdo de cada capítulo:

[11] Mais detalhes sobre a ferramenta no Capítulo 5

[12] Estudantes do curso de Ciência da Computação

[13] Desenvolvido utilizando a técnica do BDD através do BDDPM

PARTE I – CONTEXTO

CAP. 2 – A Situação do TDD e do BDD no Mercado. Discorre sobre as práticas do TDD e BDD e o cenário nas quais estão inseridas atualmente no mercado.

CAP. 3 – Ferramentas Disponíveis no Mercado e o Plugin Criado. Apresenta as principais ferramentas com foco em BDD do mercado considerando-se as seguintes plataformas WEB de gestão de projetos de desenvolvimento de software: Jira, Mantis, RedMine, Trac e BugZilla. Também apresenta o BDDPM e traz um quadro comparativo com as demais ferramentas apresentadas.

PARTE II – BDDPM EM DETALHES

CAP. 4 – O BDDPM: Mais Detalhes. Descrição detalhada de todo o processo de desenvolvimento do plugin, incluindo desafios, cronograma e etapas. Descreve a plataforma, utilitários, ferramentas, componentes e tecnologias envolvidas no desenvolvimento. Apresenta seus recursos através de exemplos de utilização e faz o enquadramento da ferramenta em uma licença de software livre.

PARTE III – AVALIAÇÃO E RESULTADOS

CAP. 5 – Aplicação do GQM e Análise dos Resultados. Descreve, de maneira detalhada, como a técnica do GQM (Goal/Question/Metric) foi aplicada para o levantamento das medições de resultados alcançados pelo produto de software.

CAP. 6 – Conclusão. Uma avaliação geral das dificuldades encontradas, dos resultados alcançados pelo produto de software, suas contribuições e um detalhamento das funcionalidades e recursos que serão implementadas em versões futuras. O capítulo é finalizado com as considerações finais a respeito do trabalho.

PARTE IV – REFERÊNCIAS BIBLIOGRÁFICAS E ANEXOS

Capítulo 2 – A Situação do TDD e do BDD no Mercado

Neste capítulo será apresentará o atual cenário do mercado em relação à Teste de Software com TDD e BDD.

A Seção 2.1 faz uma abordagem geral do TDD (Test Driven Development), seus princípios e como ele é utilizado no contexto de desenvolvimento de software. A Seção 2.2 aborda o BDD, desde de sua concepção, partindo do TDD, até os dias atuais nas fábricas de software. Finalmente, a Seção 2.3 mostra um comparativo entre as técnicas em termos de uso e aceitação no mercado, onde observa-se que a adoção do BDD chega a ser, pelo menos, 75% menor do que a do TDD.

2.1. O TDD

É creditado a Kent Beck o título de criador do TDD. Kent Beck é um Engenheiro de Software norte-americano, um dos pioneiros na área de “Padrões de Desenvolvimento de Software”[14] e “Desenvolvimento Ágil de Software”[15], e também foi um dos 17 signatários originais do “Agile Manifesto”[16].

A ideia por trás do “Desenvolvimento Dirigido por Testes” é bastante simples: os testes devem ser criados antes do código de produção. Com qual objetivo? Possibilitar que todo o código, ou a maior parte dele, possua um teste que garanta seu funcionamento. Além disso, durante a evolução do projeto de desenvolvimento de software e, fica mais fácil garantir que as alterações/incrementos não afetem a parte do sistema que já está “coberta” por testes.

Há uma corrente que afirma que Beck apenas redescobriu o TDD, uma vez que o conceito fundamental da técnica já havia sido empregado no passado e estava apenas “esquecido”. O próprio Kent confirma a teoria. Um exemplo de utilização da metodologia anterior a Kent Beck, entre outros exemplos existentes, pode ser conferido na obra “Digital Computer Programming” sob o título “Program Checkout” [*D.D. McCracken,* 1957], quando se sugeria que, para evitar problemas e erros, um cliente deveria preparar exemplos de respostas esperadas em um caso de uso de um

[14] De acordo com a Engenharia de Software, o Software Design Pattern, normalmente traduzido como "Padrão de Projeto de Software", ou "Padrão de Desenvolvimento de Software", ou ainda "Padrão de Design de Software", é uma solução genérica e reutilizável para um problema comum e recorrente em um determinado contexto no Projeto de Software. Não se trata de algo acabado, mas uma ideia, um template, uma receita para resolver um problema e que pode ser utilizada em diversas situações.

[15] Desenvolvimento ágil de software (do inglês Agile Software Development) ou Método ágil é um conjunto de metodologias de desenvolvimento de software cujo as soluções evoluem através da colaboração de times auto organizados e multifuncionais. O Desenvolvimento ágil promove um planejamento adaptativo, desenvolvimento evolucionário/incremental, entregas incrementais e antecipadas; e encoraja respostas rápidas e flexíveis a mudanças no cenário do Projeto de Desenvolvimento de Software.

[16] O Agile Manifesto, traduzido para "Manifesto Ágil" é uma declaração de princípios que fundamentam o desenvolvimento ágil de software. O manifesto contém quatro valores fundamentais e 12 princípios. Todos estão descritos no endereço http://www.agilemanifesto.org.

sistema, antes de seu desenvolvimento iniciar. Essas respostas depois seriam comparadas as saídas geradas pelo sistema.

A mecânica do TDD na prática é muito simples. Basta seguir o ciclo VERMELHO-VERDE-REFATORA/RED-GREEN-REFACTOR:

1. Escrever um teste que falha (RED/VERMELHO)
2. Fazer o teste passar da maneira mais simples possível (GREEN/VERDE)
3. Refatorar o código (REFACTOR/REFATORA).

A ideia por trás de iniciar com uma falha (1) é para garantir que o teste realmente funciona e captura um erro. Conhecendo-se o erro/problema pode-se implementar uma funcionalidade que faça a correção (2). Posteriormente a solução pode evoluir, sendo melhorada e revisada (3).

Geralmente o TDD é feito através do “Unit Testing” ou “Teste de Unidade”[17], entretanto são dois conceitos que não devem ser confundidos. O Teste de Unidade se refere ao “O QUE” está sendo testado e o TDD refere-se a “QUANDO” o teste é feito. Sendo assim, na pratica, a abordagem é criar testes de pequenas partes do código, antes de implementá-lo (TDD + Unit Testing).

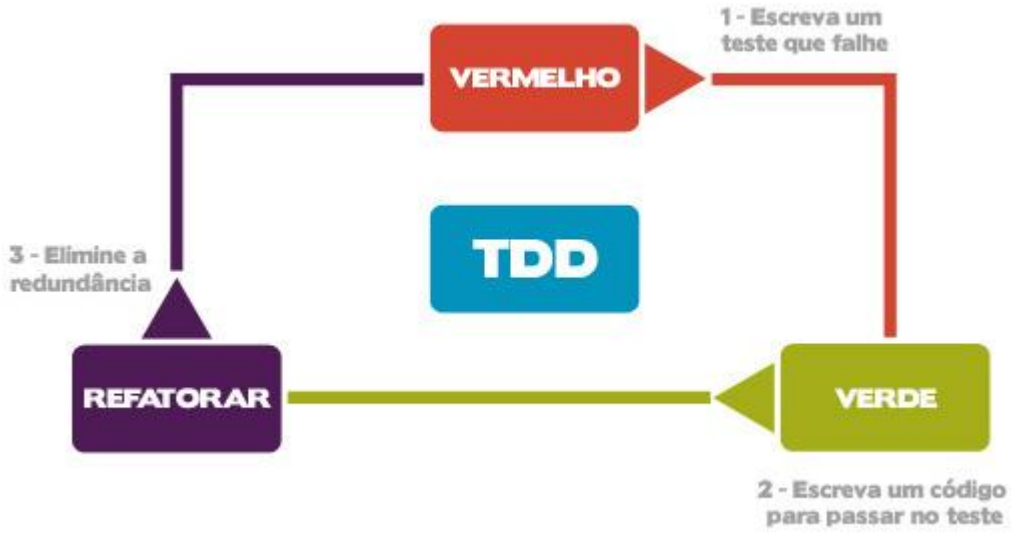

Figura 4 – Ciclo do TDD

O TDD é, atualmente, uma das mais populares práticas entre as fábricas e os desenvolvedores de software. Isso acontece por um motivo bastante simples: no mercado de software, funcionalidades que não funcionam não agregam nenhum valor ao negócio. O TDD aumenta a expectativa de funcionamento do software e, dessa maneira, provê um valor real ao negócio. O mais óbvio benefício do TDD e o mais facilmente identificado é a sua capacidade de redução de defeitos em um software.

[17] Um sistema é composto por um conjunto de unidades integradas. Uma unidade é uma parte mínima/pequena do código do sistema que agrega uma funcionalidade. Na Engenharia de Software o teste dessa pequena parte do código é chamado de "Teste de Unidade".

No artigo “Realizing quality improvement through test driven development: results and experiences of four industrial teams” [Nachiappan Nagappan; E. Michael Maximilien; Thirumalesh Bhat; Laurie Williams 2008] pesquisadores da Microsoft e da IBM, duas das maiores fábricas de software do globo, indicam que há uma redução que varia de 60% (sessenta por cento) a 90% (noventa por cento) no número de defeitos de software quando ele é submetido a abordagem do Test Driven Development (http://goo.gl/PFBHXV). Os dados foram obtidos através de análises de 04 projetos de desenvolvimento de software, sendo três na Microsoft e um na IBM.

Em um artigo mais recente, o “Test-Driven Development as an Innovation Value Chain” [Ana Paula Ress; Renato de Oliveira Moraes; Mário Sérgio Salerno 2013], realizado em uma empresa da área financeira com cerca de 3.500 (três mil e quinhentos) funcionários, obteve-se um resultado semelhante com um ganho variando de 62% (sessenta e dois por cento) a 84% (oitenta e quatro por cento). Na época foi constatado que, quanto maior o projeto e maior a familiaridade dos desenvolvedores com o TDD, maior era o benefício da prática.

Por outro lado, há uma corrente que defende que a utilização do TDD pode prejudicar, em vez de trazer benefícios. Os principais pontos criticados são:

- Perda de produtividade para programadores de nível júnior e pleno;
- Situações de difícil uso: bancos de dados, interface com usuário, funções complexas, etc.;
- Uso intenso de “Mock Objects”[18] pode induzir a falhas, uma vez que eles é quem são testados, e não os objetos reais.
- A cultura de criar testes antes de código de produção nem sempre é bem aceita por programadores e gestores;
- A aplicação tardia do TDD, ou seja, após um certo ponto do ciclo de vida do projeto de software, pode ser bastante trabalhoso. O TDD foi concebido para ser aplicado a partir dos estágios iniciais de um projeto de desenvolvimento de software;
- Os testes do TDD são, tipicamente, escritos pelos desenvolvedores. Caso haja uma falha na interpretação da especificação e/ou requisitos da funcionalidade, existirão brechas na cobertura dos testes. Normalmente os desenvolvedores possuem uma visão de criação, não voltada para testes, o que pode comprometer a escrita dos mesmos.
- A grande quantidade de testes de unidade pode trazer uma falsa sensação de segurança, impactando na diminuição de esforços em outras atividades de garantia de qualidade e segurança;
- Confusão de conceitos entre TDD e “Unit Testing”.

Em 2014 ocorreu um debate na WEB entre três grandes e reconhecidas autoridades na área de desenvolvimento de software: Kent Beck, já apresentado

[18] Objetos Mock, objetos simulados ou simplesmente Mock (do inglês Mock Object), em desenvolvimento de software, são objetos que simulam o comportamento de objetos reais de forma controlada. São normalmente criados para testar o comportamento de outros objetos. Em outras palavras, os objetos Mock são objetos “falsos” que simulam o comportamento de uma classe ou objeto “real” para que possamos focar o teste na unidade a ser testada.

anteriormente e “criador” do TDD; David Heinemeier, criador do “Ruby on Rails”[19]; e Martin Fowler, autor, conferencista e Engenheiro de Software. O debate ganhou notoriedade devido a participação dos três. A série de conversas entre eles foi intitulada “Is TDD Dead?” (O TDD está morto?), foi aberta ao público em geral e também disponibilizada na página de Martin Fowler através do endereço http://martinfowler.com/articles/is-tdd-dead nos formatos de áudio, vídeo e texto.

Tudo começou quando David expressou sua insatisfação com TDD e Testes de Unidade na comunidade do Ruby on Rails. Ele explicitou o problema da confusão entre TDD e “Unit Testing”, falou dos problemas com a utilização de Mocking, etc. Kent explicou que, nem sempre, a utilização do TDD é indicada; exemplificou com um evento promovido pelo Facebook[20], onde metade dos projetos trabalhados não permitiam uma fácil “integração” com o TDD. Kent acrescentou que, com a utilização do Test Driven Development, os programadores passam a ter mais confiança em seus códigos, porém reconheceu que a utilização de objetos Mock podem dificultar a refatoração do código, acrescentando que esta utilização pode ser evitada.

Apesar de inconclusivo, pois os 03 participantes mantiveram sua opinião no final do debate (Kent e Martin mais favoráveis ao TDD e David menos favorável), todos concordaram no ponto que, independentemente da utilização do TDD, os códigos de um sistema deveriam estar cobertos por algum tipo de “teste automático”[21], que possa ser facilmente executado ao longo do projeto e escritos de uma maneira a procurar evidenciar ao máximo falhas no código. Também concordaram que o sucesso da utilização do Test Driven Development iria variar dependendo do tipo de projeto.

2.2. O BDD

O BDD (Behavior Driven Development) é uma técnica de desenvolvimento ágil de software baseada no TDD. Ela segue, portanto, os mesmos princípios do TDD. Na verdade, o BDD é também TDD, entretanto, em vez de focar nas funcionalidades, foca no comportamento. Foi criado por Dan North com o objetivo de responder/solucionar algumas perguntas cujo TDD não era capaz de responder, ou respondia de forma incipiente. North entendeu que, para elucidar os questionamentos, fazia-se necessário uma colaboração entre todos os envolvidos no projeto do software, os stakeholders: desenvolvedores, gestores, setores de qualidade, pessoas não técnicas, etc. BDD significa “Desenvolvimento Orientado por Comportamento”, o objetivo é, literalmente, testar o comportamento esperado do software pelos stakeholders, concentrando-se nas razões pelas quais o código está sendo criado, e não em detalhes técnicos. Por

[19] Ruby on Rails, ou simplesmente Rails, é um framework Open Source para criação de aplicações web. Enfatiza a utilização de padrões de projetos e paradigmas bem conhecidos na Engenharia de Software (COC - Convention over Configuration, DRY - Don't Repeat Yourself, MVC - Model View Controller, etc.)

[20] Rede Social online lançada em 2004 e disponível através do endereço facebook.com.

[21] Automação de teste é o uso de software para controlar a execução do teste de software, a comparação dos resultados esperados com os resultados reais, a configuração das pré-condições de teste e outras funções de controle e relatório de teste. De forma geral, a automação de teste pode iniciar a partir de um processo manual de teste já estabelecido e formalizado.

esse motivo, os testes são escritos em uma linguagem de maior alto nível para facilitar o entendimento e o contato entre todos os envolvidos.

Com esta abordagem, descrita no endereço de Dan North (http://dannorth.net/introducing-bdd), as perguntas poderiam ser respondidas da seguinte maneira:

1. **Escopo do teste – o que deve e o que não deve ser testado?** Deve ser considerado como teste prioritário tudo que o cliente espera ser entregue, todos os comportamentos esperados.
2. **Como entender melhor os testes e o motivo pelos quais eles falham?** O desenvolvedor deixa a perspectiva da visão de FUNÇÕES implementadas e foca nos COMPORTAMENTOS esperados pelo cliente. Sendo assim, um teste vai falhar quando um comportamento esperado pelo cliente não for alcançado.
3. **Onde o teste inicia e onde ele termina?** Sabendo-se o desejo do cliente, basta implementar estritamente o que é desejado: nem mais, nem menos.
4. **Como denominar os testes?** Os testes passam a ser uma visão de todos os stakeholders, então, em vez de utilizar linguagens puramente técnicas, são escritos em mais alto nível com descrições claras e objetivas do que está sendo testado. Isso facilita a comunicação entre todos os envolvidos.
5. **Quanto deve ser testado?** O necessário para garantir o comportamento desejado pelo cliente.

O grande potencial do BDD encontra-se na possibilidade de escrever testes de aceitação[22] comportamental do software através de uma linguagem ubíqua[23] compartilhada entre todos envolvidos no projeto de desenvolvimento, sejam pessoas técnicas ou não. Embora não seja a linguagem oficial do BDD para criação dos testes de aceitação, o Gherkin, vem sendo amplamente utilizado para este fim devido ao grande sucesso de sua utilização no Cucumber.

Dan North indica um padrão conhecido como Given – When – Then / Dado – Quando – Então, onde um cenário de utilização do software é descrito através de uma sequência de passos. O Gherkin tem a vantagem de poder ser escrito em diversos idiomas. Atualmente são 40 línguas[24] suportadas no Cucumber.

Considerando, como exemplo, um sistema de cadastramento de estudantes em uma biblioteca central, o comportamento esperado do sistema poderia ser descrito da maneira representada na Figura 5. É uma descrição de alto nível, perfeitamente compreensível por todos os stakeholders envolvidos no projeto e que facilita a compreensão do que precisa ser testado:

[22] Teste de aceitação é uma fase do processo de teste no qual um sistema é testado antes de sua disponibilização. Tem por função verificar o sistema em relação aos seus requisitos originais, e às necessidades atuais do usuário.
[23] Quando diferentes grupos ou equipes interagem constituindo uma "interlinguagem" com conceitos, elementos e interpretações comuns a todos, linguagem esta que precisa ser poderosa o suficiente para potencializar a comunicação, facilitando o entendimento de fato e os acordos entre as partes de forma ágil e segura, com mínimo ruído. Trata-se de uma linguagem semiformal.
[24] https://github.com/cucumber/cucumber/wiki/Spoken-languages

Funcionalidade: *Cadastro de Alunos na Biblioteca Central*
Com o objetivo de possibilitar o empréstimo de livros
Como o bibliotecário
Ele deve ser capaz de cadastrar um aluno através do sistema

Cenário: *Cadastro Correto de Aluno*
Dado que um aluno esteja matriculado no 2o período, ou superior
E que ele forneça uma matrícula válida
Quando o cadastro for solicitado
Então o cadastro deverá ser aceito
Mas deve recusar, caso contrário

Figura 5 – Teste de Aceitação descrito em GHERKIN

O BDD segue os mesmos princípios do TDD e, consequentemente, o ciclo RED-GREEN-REFACTOR. A diferença é que o teste está voltado para o COMPORTAMENTO do software que está sendo desenvolvido, e a maneira como os testes são "construídos" envolve não apenas desenvolvedores e testadores, mas todos os interessados com uma grande aproximação do cliente do produto. De fato, a própria "política" do BDD indica que os clientes, em vez de se distanciarem dos testes, deveriam, eles mesmos, escrever os casos de aceitação. O gherkin permite este fim, exigindo um mínimo de conhecimento estrutural e de funcionamento da linguagem.

2.3. TDD X BDD

A comunidade BDD online é menor e menos ativa que a comunidade TDD; pode-se citar, como exemplo, o grupo oficial do BDD (https://goo.gl/AJodLS) criado por Dan North que contava, até a data de publicação deste material, com apenas 604 (seiscentos e quatro) membros, enquanto que um simples grupo não oficial do TDD (https://goo.gl/E4Kqyi) ultrapassa esta marca. Acrescenta-se que, uma pesquisa por "TEST DRIVEN DEVELOPMENT" no Google (www.google.com) retorna, aproximadamente, 703.000 (setecentos e três mil) resultados enquanto que a pesquisa por "BEHAVIOR DRIVEN DEVELOPMENT" retorna, aproximadamente, 177.000 (cento e setenta e sete mil) resultados. O comparativo (ver Tabela 2) também foi feito no Bing (www.bing.com) e no Yahoo (www.yahoo.com), incluindo-se assim os resultados obtidos de 03 importantes e reconhecidas ferramentas de busca de nível global. Não foi feita uma comparação direta entre os termos TDD e BDD, pois BDD também é um acrônimo para termos bastante comuns na língua inglesa (*Binary Decision Diagram, Body Dysmorphic Disorder, Business Driven Development*, etc.), o que traria erro nos resultados. A diferença também foi evidenciada na pesquisa por artigos acadêmicos (IEEE[25], Google Scholar[26], ACM[27], CitesserX[28], Capes[29]). Ver Tabela 3.

[25] https://www.ieee.org
[26] http://scholar.google.com
[27] http://dl.acm.org
[28] http://citeseer.ist.psu.edu
[29] http://www.periodicos.capes.gov.br

Aliando estes dados às informações obtidas do ambiente empresarial atual, conclui-se que, em relação a utilização e participação no mercado, o BDD ainda se apresenta de maneira incipiente em comparação ao TDD, gerando pelo menos 75% (setenta e cinco por cento) a menos de resultados. Todas as pesquisas foram realizadas em dezembro de 2014.

Sistema de Busca	Qtde. de Resultados - TDD	Qtde. de Resultados - BDD
GOOGLE	703.000	177.000
BING	1.040.000	63.000
YAHOO	1.000.000	62.800

Tabela 2 – Comparativo de retorno de resultado nos principais buscadores WEB (TDD x BDD)

Sistema de Busca	Qtde. de Resultados - TDD	Qtde. de Resultados - BDD
IEEE	407.000	34.300
GOOGLE SCHOLAR	11.300	538
ACM Digital Library	1.448	62
CITESEERX	1.745	47
ACM	309	8
PERIÓDICOS/CAPES	143	7

Tabela 3 – Comparativo de retorno de resultado acadêmico (TDD x BDD)

A forma como o TDD e o BDD são aplicados dentro das empresas, na prática, varia bastante. Alguns dos fatores que afetam esta utilização são:

- **Porte dos Projetos** – Estudos mostram que o TDD obtém melhores resultados em projetos de grande porte [Nachiappan Nagappan et al, 2008]. Por este motivo as empresas, ou não aplicam TDD nos projetos pequenos, ou “enxugam” o TDD para que ele possa ser aplicado nesses projetos.
- **Cultura da Empresa e Objetivos da Empresa** – Algumas empresas não possuem a cultura de testar o software desenvolvido por elas; o motivo para isso é bastante variado. No caso da PBprev, por exemplo, o TDD e/ou o BDD foram aplicados em apenas 50% dos projetos de desenvolvimento de software nos últimos 02 anos. Todos de maneira superficial. A justificativa é que a empresa não é voltada para venda de software, eles são criados para uso interno e a alta gestão alega que não há a necessidade de se investir em uma equipe de testes, nem em tempo e esforços em atividades relacionadas à qualidade de software. Além disso os projetos são pequenos e de baixa complexidade.
- **Experiência e Opinião de Colaboradores** – Não há consenso até mesmo entre especialistas na área. David Heinemeier, criador do “Ruby on Rails”, prega que o TDD está morto. Kent Beck vê potencial na técnica. Como colaboradores de empresas, essas visões afetam como eles trabalham e, consequentemente, na forma como os métodos são utilizados dentro das empresas.

Em muitas empresas o BDD, ou não é adotado, ou é colocado em segundo plano em detrimento do TDD. Na PBprev, por exemplo, o número de projetos que utilizam TDD é 03 vezes maior que o de projetos que utilizam BDD. Grande parte disso se

deve a, já discutida, cultura amplamente adotada na qual as empresas esperam entregar um sistema livre de defeitos sem se preocupar com a entrega de valor ao cliente, ou seja, um produto que atende as expectativas de comportamento esperadas pelo cliente.

Existe um ponto onde este problema inicia: a escrita dos testes. Quem deveria escrever os testes, para que o BDD atingisse todo o seu potencial, deveriam ser os clientes. BDD objetiva aproximar os clientes (parte não técnica) do pessoal técnico envolvido no desenvolvimento do projeto. Como, em muitos casos, o contato com o cliente do produto nem sempre é fácil e frequente, os próprios desenvolvedores e/ou testadores acabam escrevendo o código na visão de funcionalidades e acabam por voltar ao TDD.

O BDDPM visa facilitar a adoção do BDD concentrando esforços no ponto da falha. Ele foi concebido para que os próprios clientes façam a escrita dos testes e possam acompanhar, eles mesmos, os resultados de maneira remota e através de uma interface simples e amigável, e deixando o Gherkin transparente, pois, ainda que o Gherkin seja uma linguagem fácil e de alto nível, ela pode se tornar um entrave para clientes que não entendam sua sintaxe e/ou seu padrão. O BDDPM procura extrair do cliente o que importa, seu conhecimento de negócio, sem expô-lo aos conceitos de BDD e, ainda assim, aplicando BDD.

A partir do próximo capítulo o foco deste trabalho será o BDD, quais plugins existem nos mercados para se trabalhar com projetos de desenvolvimento de software, como o BDDPM auxilia nesse processo e como BDDPM foi avaliado considerando-se seus objetivos.

Capítulo 3 – Ferramentas Disponíveis no Mercado e o Plugin Criado

O objetivo deste capítulo é mostrar, de maneira geral e sucinta, quais extensões/plugins BDD estão disponíveis no mercado para utilização em conjunto com plataformas WEB de gestão de projetos/falhas de software e, com isso, evidenciar a escassez desses produtos quando focam na técnica do Behavior Driven Development.

Embora existam no mercado diversas ferramentas stand-alone[30] para trabalho com o Behavior Driven Development, o BDDPM adota a abordagem de extensão/plugin, ou seja, é necessário que exista uma plataforma por baixo para que a ferramenta funcione. Conforme especificado nos capítulos anteriores, o BDDPM visa impulsionar a utilização do BDD no contexto de projetos de desenvolvimento de software. Nesse sentido, uma plataforma de gestão de projetos poderia prover todos os recursos voltados a estes tipos de projetos, enquanto o BDDPM adicionaria os recursos BDD. As 02 partes seriam beneficiadas: o BDDPM não precisaria se preocupar em relação aos recursos gerenciais e, ao mesmo tempo, atuaria como ferramenta de apoio à plataforma, fornecendo funcionalidades para testes orientados por comportamento, com impacto positivo nos índices de sucesso dos projetos geridos por ela; uma vez que estudos indicam que pode haver um ganho de até 35% no tempo de desenvolvimento de software, quando boas práticas de testes são adotadas [Nachiappan Nagappan et al, 2008]. Sendo assim, o BDDPM entra na categoria de plugin. As principais plataformas de gestão de projetos do mercado, incluindo à plataforma sob a qual o BDDPM irá atuar, serão apresentadas neste capítulo.

A primeira Seção deste capítulo, a 3.1, inicia fazendo uma apresentação das seguintes plataformas: Jira, Mantis, RedMine, Trac e o BugZilla, que são as mais tradicionais no mercado. Também descreve, quando existir, os plugins BDD para cada uma. A Seção 3.2 elenca os motivos para escolha do Mantis como plataforma para criação do BDDPM. Para finalizar, na Seção 3.3 faz uma apresentação do plugin, fruto deste trabalho; a 3.4 faz um comparativo deste plugin com outras ferramentas encontradas no mercado e a 3.5 descreve a escolha da licença que rege o plugin.

3.1. Plataformas de Gestão de Projetos Online

As plataformas escolhidas para avaliação e estudo deste trabalho são: Jira, Mantis, RedMine, Trac e BugZilla. Os critérios para escolha dos mesmos foram: (1) popularidade[31] e (2) tempo de mercado[32].

[30] Programas autossuficientes, ou seja, para seus funcionamentos não necessitam de um software auxiliar.
[31] http://goo.gl/RG7B18
[32] http://goo.gl/DWN6Wc

As 05 ferramentas estão no TOP 10 entre as mais utilizadas atualmente. O Mantis tem 15 anos de mercado e o Jira, RedMine, Trac e BugZilla contam com, respectivamente (em anos): 13, 9, 9 e 17; tratam-se, portanto, de plataformas consolidadas. Muito embora algumas delas sejam formalmente chamadas de sistemas de acompanhamento de bugs[33], na prática elas também são aproveitadas para: cadastro de atividades para desenvolvedores e testadores, cadastro de funcionalidades do produto, comunicações referentes ao projeto, registro de horas trabalhadas, entre outros. Com isso, e através do acompanhamento desses registros, elas acabam por ser utilizadas para fazer o acompanhamento do projeto de maneira geral, e não apenas para gestão de falhas. Algumas delas já se intitulam como ferramentas de gestão de projetos.

3.1.1. Sobre o Jira

O Jira é uma plataforma proprietária da Atlassian[34] para gestão de projetos. De acordo com a Atlassian, o Jira é utilizado por mais de 25.000 clientes em 122 países, inclusive por grandes empresas como a Nasa[35], Adobe[36], BMW[37], Cisco[38], etc. Em relação aos plugins que adicionam o suporte ao BDD ao Jira, o repositório oficial da Atlassian, o Atlassian Market, lista 02 ferramentas:

- **Behave for Jira** – Ferramenta paga para integrar o Jira ao Cucumber, SpecFlow e outras ferramentas para automação de testes. Permite a criação de testes de aceitação BDD. O repositório do plugin (http://goo.gl/BZL5kK) indica que ele recebeu, desde seu lançamento, 06 avaliações, recebendo nota 2.5 de um total de 4. O Behave permite a adição de Cenários BDD escritos na linguagem Gherkin. Os testes são executados através da integração com as ferramentas de automação de testes. Algumas telas do produto podem ser conferidas através das Figuras 11, 12 e 13. É uma ferramenta paga da Hindsight[39]. O site oficial da ferramenta é o http://hindsightsoftware.com/solutions/behave-for-jira. O custo de uso varia dos $ 100,00/ano para 10 usuários até $1.200,00/ano para mais de 10.000 usuários.
- **Cucumber Report Plugin** – Plugin gratuito que permite a integração com o Cucumber apenas para geração de relatórios de resultados de testes de comportamento, um front-end mais amigável para os resultados gerados no Cucumber. A página oficial do plugin indica que a ferramenta recebeu 3 avaliações, atingindo a nota 3 de 4 (http://goo.gl/wYzcgb). Este plugin é distribuído pela mesma empresa do plugin anterior, a Hindsight, e normalmente é utilizado em conjunto com aquele nos projetos cadastrados no Jira para adição de suporte ao BDD mais a possibilidade

[33] Uma falha, um problema, um defeito de software é comumente chamada de bug de software.
[34] https://www.atlassian.com/software/jira
[35] http://www.nasa.gov
[36] http://www.adobe.com
[37] http://www.bmw.com
[38] http://www.cisco.com
[39] http://hindsightsoftware.com

de acompanhar os resultados dos testes através de relatórios do Cucumber dentro da interface do Jira.

3.1.2. Sobre o Redmine

O Redmine é uma ferramenta WEB escrita em Ruby on Rails, gratuita e de código livre (Open Source) sob a licença GNU General Public License v2 (GPL[40]). Objetiva facilitar a gestão de projetos e o controle de falhas. Entre as diversas entidades que utilizam o produto, encontram-se também órgãos governamentais como a Iniciativa de Código Livre do Chile (http://www.softwarepublico.cl), o Ministério de Educação da França (http://dev-eole.ac-dijon.fr) e o Departamento de Energia dos EUA (https://cdcvs.fnal.gov/redmine). Em relação aos plugins que adicionam o suporte a BDD ao Redmine, o repositório oficial da Plugins, disponível através do endereço http://www.redmine.org/plugins, não traz resultados. O site oficial do Redmine pode ser acessado através do endereço http://www.redmine.org.

3.1.3. Sobre o Trac

O Trac é uma ferramenta simplificada para acompanhamento de problemas gestão de projetos de desenvolvimento de software. Trata-se de uma ferramenta WIKI voltada para este fim. Utiliza uma abordagem minimalista estimulando o contato colaborativo entre todos os membros da equipe. Foi desenvolvido na linguagem de programação Python. Inicialmente era distribuído sob a licença GPL, porém, desde a versão 0.9, é disponibilizado sob uma licença BSD[41] modificada. O Trac é utilizado em vários projetos reconhecidos na WEB (Django[42], Tor[43], Jquery[44], etc.). O site oficial da ferramenta é o http://trac.edgewall.org. O repositório de plugins do Trac é acessível através do endereço eletrônico http://trac.edgewall.org/wiki/PluginList e seu site oficial através do endereço http://trac.edgewall.org. Em relação à plugins de testes, o repositório oficial lista 02 resultados (“Test Manager Plugin for Trac”[45] e “Test Case

[40] A GNU General Public License, GNU GPL, ou simplesmente GPL, é a designação da licença para software livre idealizada por Richard Matthew Stallman em 1989, no âmbito do projeto GNU da Free Software Foundation (FSF). GPL é a licença com maior utilização por parte de projetos de software livre. A GPL visa garantir que qualquer produto de software (regido por suas regras) possa ser compartilhado e modificado por qualquer pessoa e permaneça assim, indefinidamente. O software pode ser vendido, mas deve ser sempre aberto. A versão 2 da licença prega que qualquer tentativa de impedir esta liberdade acarreta na impossibilidade de distribuição do software modificado.

[41] A licença BSD é uma licença de código aberto inicialmente utilizada nos sistemas operacionais do tipo Berkeley Software Distribution (um sistema derivado do Unix). A licença BSD estabelece que o detentor do copyright cede os direitos comerciais, mas exige crédito pela autoria e propriedade. A redistribuições do código fonte devem manter a notícia de copyright, as condições da licença e um aviso de que não há garantias nem se assume a responsabilidade por prejuízos decorrentes do uso do software. Distribuições binárias devem reproduzir na documentação essas informações. Os nomes do autor e seus colaboradores não podem ser usados para endossar ou promover produtos derivados sem permissão.

[42] https://code.djangoproject.com

[43] https://trac.torproject.org/projects/tor

[44] http://bugs.jquery.com

[45] http://trac-hacks.org/wiki/TestManagerForTracPlugin

Management Plugin"[46]), nenhum deles especificamente voltados para BDD, ou seja, suas utilizações deveriam ser adaptadas para atingirem este fim.

3.1.4. Sobre o BugZilla

O BugZilla é uma ferramenta de software desenvolvido para auxílio na gestão de projetos de desenvolvimento de software fornecendo um sistema de rastreamento de defeitos. É uma ferramenta consolidada no mercado, com mais de 17 anos de história, e é um projeto de software livre, sendo mantido por voluntários e constantemente testado e utilizado pela Fundação Mozilla[47]. Mais de 148[48] empresas ao redor do globo utilizam o BugZilla, que é utilizado em projetos notoriamente conhecidos na área de TI: Mozilla[49], Linux Kernel[50], GNOME[51], KDE[52], Apache Project[53], Libre Office[54], Open Office[55], Eclipse[56], Red Hat[57], Mandriva[58], Gentoo[59], Novell[60], etc. O site oficial do BugZilla é acessível através da URL https://www.bugzilla.org. A lista de plugins e extensões disponíveis para a plataforma podem ser conferidas através do endereço oficial https://wiki.mozilla.org/Bugzilla:Addons. São listados 11 ferramentas que podem ser utilizadas em conjunto com o BugZilla para atividades que envolvem testes de software. Nenhuma delas tem foco em BDD.

3.1.5. Sobre o Mantis

A plataforma escolhida para o desenvolvimento do BDDPM foi o Mantis. O Mantis Bug Tracker é um aplicativo WEB gratuito e open source para gestão e controle de falhas. É distribuído sob a licença GPL V2 e desenvolvido com PHP. Embora o MantisBT seja bastante utilizado para controle de defeitos em produtos de software, também é utilizado como ferramenta de gestão de projetos. O nome Mantis e a sua logomarca referem-se a uma família de insetos que costuma rastrear e se alimentar de outros insetos (bugs), daí o nome Bug Tracker (MantisBT). O projeto do Mantis foi iniciado no ano de 2000. O site oficial da ferramenta pode ser acessado através do endereço https://www.mantisbt.org. O canal oficial de plugins do Mantis está disponível através do endereço https://github.com/mantisbt-plugins. Até a data de publicação deste trabalho, o diretório não listava plugins de testes e, consequentemente, nenhum plugin/extensão para trabalho com Behavior Driven Development.

---

[46] http://trac-hacks.org/wiki/TestCaseManagementPlugin
[47] https://www.mozilla.org
[48] https://www.bugzilla.org/installation-list
[49] https://bugzilla.mozilla.org
[50] http://bugzilla.kernel.org
[51] http://bugzilla.gnome.org
[52] http://bugs.kde.org
[53] http://issues.apache.org/bugzilla
[54] http://bugs.libreoffice.org
[55] http://www.openoffice.org/issues/query.cgi
[56] http://bugs.eclipse.org/bugs
[57] https://bugzilla.redhat.com/bugzilla
[58] http://qa.mandriva.com
[59] http://bugs.gentoo.org
[60] https://bugzilla.novell.com

3.2. A Escolha da Plataforma

Facilitar a adoção do BDD no mercado, um dos principais objetivos do BDDPM. Qualquer uma das plataformas citadas anteriormente poderia servir como base para o novo plugin criado. De fato, se o objetivo é a adoção da técnica do Behavior Driven Development, então o BDDPM deveria estar disponível para todas as plataformas, pois, com isso, atingiria uma comunidade maior de usuários, potencializando os objetivos. Contudo, para minimizar o escopo de trabalho, uma plataforma foi escolhida. É importante ressaltar, entretanto, que, embora o BDDPM foi desenvolvido com grandes partes das funcionalidades feitas em Java Script e rodando do lado do cliente/navegador. Essa estratégia facilitará a portabilidade para outras plataformas no futuro.

Alguns critérios foram utilizados para escolha da plataforma:

- **A plataforma deveria estar em uma linguagem conhecida pela equipe de desenvolvimento do BDDPM:** com o objetivo de acelerar o desenvolvimento do plugin, uma maior familiaridade da equipe de desenvolvimento com a linguagem de programação da plataforma, impulsionaria os resultados. A equipe do projeto tem experiência com C#, Java e PHP.
- **A plataforma deveria ter poucos, ou nenhum, ferramental BDD disponível no mercado:** se o objetivo é facilitar a adoção, procurar por um setor ainda não abastecido por recursos torna-se uma estratégia interessante. No caso de empate, seria escolhida a plataforma com menos opções gratuitas de plugins, visto que alguns deles são comerciais e precisam ser comprados.

O Jira permite a criação de plugins e Java, porém já possui alguns componentes disponíveis do mercado, uns pagos, outros não. O Redmine não lista plugins BDD disponíveis em sua página de repositórios, porém utiliza a linguagem Ruby. O Trac possui alguns plugins voltados para testes de software, porém foi escrito em Python. O BugZilla segue o mesmo caminho do Trac, porém é escrito em Perl. O Mantis é a única plataforma que passa nos 02 critérios estabelecidos: é uma plataforma que utiliza PHP e não possui extensões BDD, por este motivo foi a plataforma escolhida para criação do BDDPM.

3.3. O Plugin Criado

O BDD Plugin for Mantis (BDDPM) foi criado utilizando PHP no back-end e HTML5, CSS e Java Script (JS) no front-end. O SGBD[61] (Sistema de Gerenciamento de Banco de Dados) utilizado é o padrão do Mantis: o MySQL. Toda parte da lógica de negócios da ferramenta foi desenvolvida com JS para facilitar o porte do plugin para outras plataformas como o Jira, Mantis, RedMine, Trac, BugZilla, etc. Na parte do servidor foi colocada apenas a lógica para manipulação de arquivos e integração

[61] Um Sistema de Gerenciamento de Banco de Dados (SGBD) - do inglês Data Base Management System (DBMS) - é o conjunto de programas de computador (software) responsáveis pelo gerenciamento de uma base de dados.

com o repositório do projeto. Algumas métricas do projeto completo (incluindo componentes de terceiros):

- 64 arquivos de código
- 7259 linhas de código
- 669 linhas de comentário

Ainda do ponto de vista técnico, o BDDPM, além do código próprio, foi construído utilizando recursos de frameworks, extensões e componentes de terceiros:

- PHP GIT Library (github.com/kbjr/Git.php)
- PHP ORM Library (taq.github.io/torm)
- AngularJS (angularjs.org)
- StringJS (stringjs.com)
- JQuery (jquery.com)
- JQuery BlockUI Plugin (malsup.com/jquery/block)
- JQuery Colorbox Plugin (jacklmoore.com/colorbox)
- JQuery qTip² Plugin (qtip2.com)
- JQuery Sweet Alert Plugin (tristanedwards.me/sweetalert)
- JQueryUI (jqueryui.com)
- Metro UI CSS (metroui.org.ua)

O objetivo do BDDPM é adicionar recursos ao Mantis de maneira a possibilitar a utilização do BDD em projetos de desenvolvimento de software. Espera-se agregar valor para gerentes de projeto, com mais uma opção/técnica/metodologia de realização de testes; para os clientes do projeto de software, com as opções de inclusão de testes por eles mesmos em linguagem simples e de alto nível e a de acompanhamento dos testes em tempo real de forma remota; para os desenvolvedores do projeto com a geração automática dos códigos de teste e de comunicação com o plugin; e para os testadores do software com a criação automática de rotinas de testes.

O cliente participa de todo o processo descrevendo o comportamento desejado do produto de software através de um formulário dinâmico onde ele estabelece o comportamento para a funcionalidade desejada, dando um título para ela, uma breve descrição e um conjunto de passos no formato Given-When-Then (Dado que..., quando..., então...). Essas informações são, então, transformadas em uma linguagem que é utilizada em ferramentas BDD, o Gherkin[62]. Basicamente é uma linguagem técnica de alto nível, que descreve passos, seguindo um conjunto de regras e que é facilmente processada por ferramentas BDD para realizar os testes de comportamento do software, ou seja, é a linguagem de comunicação intermediária entre a descrição do comportamento que o cliente quer e a ferramenta que testa estes comportamentos.

Tanto o arquivo com a descrição do comportamento em Gherkin, quanto os códigos de testes gerados em C#.NET, são salvos no repositório do projeto de software. Os desenvolvedores, então, terão acesso a estes códigos através de um

[62] O Gherkin é a linguagem utilizada em ferramentas que trabalham com o BDD. É uma linguagem de negócios que permite descrever o comportamento do software sem detalhar como esse comportamento é implementado.

ambiente de desenvolvimento configurado para o projeto: tipicamente o "Visual Studio"[63]. Para que os testes possam ser executados a partir da IDE[64], é necessário um componente: o SpecFlow, uma implementação do Cucumber para a plataforma .NET da Microsoft. Este componente é quem entende o Gherkin e executa os testes criados especificamente para eles. Trata-se de um requisito OBRIGATÓRIO para o funcionamento do BDD Plugin for Mantis, cujo repositório encontra-se no endereço "https://alvaromagnum@bitbucket.org/alvaromagnum/bddpm.git" e o SpecFlow pode ser baixado através do "http://www.specflow.org".

Durante a criação dos testes, o cliente utiliza uma interface, que segue o padrão Wizard[65], para escrever os testes de aceitação no padrão GIVEN-WHEN-THEN. Essa entrada do usuário é, posteriormente, convertida em Gherkin e enviada ao repositório GIT[66] do projeto. Dentre os recursos disponíveis no Gherkin, estes são atualmente suportados pelo BDDPM: Feature, Scenario, Given, When, Then, And, But. Posteriormente os seguintes recursos serão incluídos: Background, Transformations, Examples, Data Tables e suporte à múltiplos idiomas. Segue a descrição dos recursos:

- **Feature** – Descreve a funcionalidade que será testada.
- **Scenario** – Descreve um caso de uso para a funcionalidade testada.
- **Given** – Descreve uma condição/fato de teste.
- **When** – Descreve um evento de teste.
- **Then** – Descreve uma consequência de teste.
- **And** – Descreve uma adição de condição/fato de teste.
- **But** – Descreve uma ressalva de condição/fato de teste.
- **Background*** – Descreve uma situação verdadeira para vários cenários de teste.
- **Transformations*** – Permite o trabalho com vários tipos de dados de entrada de testes (datas, números, caracteres, valores monetários, valores lógicos, etc.).
- **Examples** – Permite o trabalho com diversas entradas de dados de testes em um cenário.
- **Data Tables*** – Permite o trabalho com diversas entradas de dados de testes em uma condição/fato de teste.

[63] O Visual Studio é um ambiente de desenvolvimento integrado (IDE) da Microsoft. Ele é usado para desenvolver programas de computador para o Windows, bem como web sites, aplicações web e serviços web. O Visual Studio é capaz de construir software para diversas plataformas Microsoft como o Windows Azure, Windows Forms, Windows Presentation Foundation, Windows Store, Microsoft Silverlight, etc.

[64] IDE, do inglês Integrated Development Environment ou Ambiente Integrado de Desenvolvimento, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.

[65] Assistente ou Wizard é um padrão de projeto de software amplamente utilizado em interface gráfica do usuário para prover um meio simples de realizar tarefas complexas em sistemas computacionais, através de um esquema passo-a-passo.

[66] GIT é um sistema de controle de versão distribuído e um sistema de gerenciamento de código fonte. O controle de versão é um sistema que registra as mudanças feitas em um arquivo ou um conjunto de arquivos ao longo do tempo de forma que se possa recuperar versões específicas.

* Recursos a serem adicionados posteriormente.

Os “Step Definitions” são gerados automaticamente na linguagem C#.NET (posteriormente será adicionado suporte ao JAVA) e salvos no repositório do projeto. Os resultados dos testes são acompanhados em tempo real pelo cliente a medida em que vão sendo executados. Todas as atividades podem ser realizadas remotamente, através da WEB, o que facilita o contato entre os envolvidos e possibilita um rápido feedback[67] para o cliente; o que faz parte da cultura de fortalecimento de intercomunicação, pregado pelo BDD, entre os stakeholders do projeto.

De maneira resumida, este é o fluxo de funcionamento do BDDPM:

1. O Gerente do Projeto cria o projeto de desenvolvimento de software no Mantis.
2. O Gerente e/ou o Arquiteto do Projeto configuram o projeto adicionando os dados de acesso ao repositório GIT (usuário, senha e URL/Caminho Local).
3. Os clientes do projeto descrevem, em alto nível, o comportamento desejado do software.
4. A descrição do cliente é transformada em códigos e rotinas de testes para desenvolvedores e testadores. Os códigos são adicionados automaticamente ao repositório do projeto.
5. Desenvolvedores conectados ao repositório do projeto completam o código gerado para os testes para fazê-los “passar”[68].
6. Os testadores irão executar as rotinas de testes em horários estabelecidos pelo Gerente do Projeto. As rotinas geradas pelo BDDPM irão se comunicar automaticamente com a ferramenta para atualizar o Banco de Dados com os resultados obtidos nos testes.
7. O cliente, por fim, poderá acompanhar os resultados dos testes em tempo real, pois os resultados são gerados sempre que os testes são executados. O cliente saberá quais testes ainda não foram executados, quais estão falhando e quais estão passando.

3.4. Comparativo

O BDDPM, conforme já explanado, é um plugin/extensão para o MantisBT que passa a estar disponível no mercado a partir da publicação deste trabalho. A ferramenta é distribuída sobe a licença GPL. A Tabela 4 exibe um comparativo entre o Behave, plugin para o JIRA, e o BDDPM. O Behave foi a única extensão encontrada para adição de suporte ao Behavior Driven Development considerando-se as plataformas elencadas anteriormente.

[67] O feedback, retorno de informação ou retorno é o procedimento que consiste no provimento de informação a uma pessoa sobre desempenho, conduta, ação ou resultado de algo ou alguém.

[68] No fluxo de testes do TDD, do qual o BDD deriva, o desenvolvedor espera um teste falhar antes de fazer o código par que o teste funcione da maneira correta. Essa estratégia é chamada de Red-Green-Refactor. Quando o teste falha, o desenvolvedor obtém uma Barra Vermelha (Red). O programador refaz os testes até que eles funcionem da maneira correta e obtém uma Barra Verde (Green), depois é só refatorar o código para que se tenha, além de algo funcional, um código de qualidade.

RECURSO	BDDPM	BEHAVE
Ferramenta para Automação	SpecFlow	Cucumber[69]
Forma de Entrada de Testes de Aceitação	Gherkin/Wizard	Gherkin/Texto
Preço	Gratuito	Pago
Geração de Código de Teste	SIM	NÃO
Linguagem de Programação Suportada	C#	VÁRIAS[70]
Suporte a Todos os Recursos do Gherkin	NÃO[71]	SIM
Acompanhamento em Tempo Real	SIM	NÃO[72]
Suporte à Comentários e Anexos	NÃO[73]	SIM

Tabela 4 – Comparativo entre BDDPM (Mantis) e BEHAVE (Jira)

- **Ferramenta para Automação do Testes:** esta é a ferramenta que irá, de fato, processar e executar os testes criados no plugin. O Behave utiliza o Cucumber diretamente, o BDDPM utiliza uma implementação do Cucumber para a plataforma .NET da Microsoft: o SpecFlow. O SpecFlow é compatível com o Cucumber e é listado na página oficial do Cucumber (https://cukes.info/docs#cucumber-implementations).
- **Forma de Entrada de Testes de Aceitação:** descreve a forma como os testes são criados, trabalhados e inseridos no projeto. Tanto o BDDPM como o Behave baseiam-se no Gherkin, uma vez que tudo baseia-se no Cucumber. A diferença é que no Behave, é necessário conhecer o Gherkin e escrever os testes direto na linguagem. No BDDPM isso não é necessário, o usuário detalha seus testes através de um Wizard que irá fazer a montagem do Gherkin em Background.
- **Preço:** indica se é necessário pagar algum valor para utilizar a ferramenta. O Behave requer o pagamento de uma licença de uso que varia de acordo com a quantidade de usuários da ferramenta. O BDDPM é uma solução de código aberto completamente gratuita.
- **Geração de Código de Testes:** indica se a ferramenta é capaz de fazer a geração dos códigos de testes em alguma linguagem de programação. O Behave permite apenas a criação/inclusão dos testes. A criação dos códigos para testar o software do projeto fica a cargo do(s) desenvolvedor(es) do projeto. O BDDPM atua de maneira diferente, ele é capaz de gerar códigos de programação em C# com a base necessária para que os testes possam ser implementados e executados. Além disso, o código criado apresenta uma rotina de comunicação com o BDDPM para envio de status dos testes.

[69] Há suporte para exportação para o SpecFlow e Behat

[70] Como não há a geração de código de teste, o Behave é independente de linguagem. Os testes serão executados em cima do projeto cadastrado no Jira. Os projetos podem ser em qualquer linguagem.

[71] O Gherkin Wizard do BDDPM ainda não permite o trabalho com todos os recursos do Gherkin. O Behave, por possuir uma entrada em formato de texto livre, aceita todas as definições Gherkin suportadas pelo Cucumber. O suporte completo às funcionalidades do Gherkin será incorporado em uma versão futura do BDDPM.

[72] O BDDPM é atualizado à medida que cada teste é rodado, a cada execução. No Behave os resultados dos testes são acompanhados externamente, no Cucumber. Para acompanhamento do relatório de testes, o Jira faz uso de outra ferramenta, o Cucumber Report Plugin.

[73] Estes recursos serão adicionados em uma versão futura.

- **Linguagem de Programação Suportada:** identifica qual(is) a(s) linguagem(ens) de programação suportada(s) pelas ferramentas. O Behave é executado em cima do Jira que é uma plataforma de gestão de projetos de software independente de linguagem. A mesma coisa acontece com o BDDPM, que executa em cima do Mantis. A diferença é que o BDDPM é voltado exclusivamente para projetos desenvolvidos em C# e é capaz de gerar códigos de teste nessa linguagem. O Behave serve para projetos em qualquer linguagem, entretanto não gera código de testes em nenhuma delas.
- **Suporte a todos os recursos do Gherkin:** indica se a solução implementa todos os recursos suportados pelo Gherkin e, consequentemente, todos os recursos possibilitados pelo BDD via Cucumber ou compatível. O BDDPM trabalha apenas com as funcionalidades básicas do BDD, visto que é um projeto piloto[74]. O Behave implementa todas as funcionalidades e permite escrever os testes diretamente em Gherkin.
- **Acompanhamento em Tempo Real dos Testes:** indica se a ferramenta é capaz de prover relatórios em tempo real a respeito do status dos testes executados. O BDDPM possui esta funcionalidade uma vez que é capaz de gerar códigos de testes com rotinas de comunicação que enviam estas informações para o BDDPM sempre que um teste é executado. O Behave não apresenta esta característica, o acompanhamento dos testes deve ser feito via Cucumber.
- **Suporte à Comentários e Anexos:** indica se a ferramenta dá suporte à inserção de informações auxiliares aos testes criados, como comentários e anexos. O BDDPM não apresenta esta funcionalidade que é implementada no Behave.

O BDDPM foca na facilidade de uso com o objetivo de induzir os próprios clientes do projeto de software a escreverem os testes de aceitação, partindo do princípio de que eles não precisam entender os conceitos e/ou estrutura do Gherkin. O cenário desejado de utilização da ferramenta é o cenário no qual apenas os clientes criam os casos de uso das aplicações, os desenvolvedores e testadores se preocupam no desenvolvimento e nos testes, e o gestor do projeto faz o acompanhamento com o auxílio dos índices exibidos pelo BDDPM.

O BDDPM não tem o objetivo de ser uma ferramenta exaustiva sobre o Behavior Driven Development, mas algo que traz o mínimo necessário para aplicação do BDD com foco no incentivo à adoção da técnica. O resultado deste esforço é o fator determinante para o futuro do BDDPM. Nos próximos dois capítulos será feita a avaliação do plugin quanto ao atingimento das metas, a avaliação dos resultados e serão detalhados os próximos passos da ferramenta.

[74] A expressão projeto piloto costuma se referir a qualquer plano/projeto preliminar ou de embasamento a um empreendimento. Pode-se tratar de uma primeira versão de algo, uma proposta preliminar, etc.

3.5. Licença do BDDPM

Conforme visto anteriormente, o BDDPM faz uso de componentes de terceiros que são distribuídos apenas sob duas licenças (ou combinação delas): a GPL[75] e a MIT[76]. A licença MIT permite que o código regido por ela possa ser modificado e/ou incluído em um outro produto e que o resultado obtido pode ser distribuído novamente em formato de software livre ou proprietário; inclusive o produto final pode ser comercializado e ter o código fechado. É uma licença permissiva, pois impõe poucas restrições. Por outro lado, a GPL impede que o código não seja aberto. Obrigatoriamente, todo software disponibilizado sob a GPL deve ter seu código aberto e livre para ser alterado por qualquer pessoa ou organização.

De acordo com a FSF[77] (Free Software Foundation), toda licença compatível com a GPL permite que os códigos regidos por elas possam ser combinados com códigos sob a GPL, desde que o resultado final desta combinação, se for distribuído, deve ser distribuído sob a licença GPL[78]. A licença MIT é compatível[79] com a GPL.

O BDDPM foi desenvolvido em um ambiente acadêmico que, por natureza, é tipicamente colaborativo. O objetivo da ferramenta é facilitar/estimular a adoção de BDD no mercado. Contribuições para o projeto são esperadas e estimuladas. Cada evolução da ferramenta, caso venha a ser compartilhada, também deve ter o código aberto para que o processo de melhoria contínua seja viável a todo tempo. O criador do BDDPM entende que se algo foi criado com um objetivo bem definido e esse algo pode ser melhorado, então seu objetivo também será afetado e terá maior impacto no que propõe. O próprio BDDPM se beneficiou desse pensamento ao utilizar componentes licenciados sob a GPL. Por estes motivos, o BDDPM será distribuído sob a licença GPL v3, a versão mais recente, pois ela estabelece uma série de princípios (de maneira mais explícita que a versão 2) que coíbe a modificação e/ou distribuição de software sob a GPL com restrições aos usuários, por mal interpretação da GPL v2; qualquer tentativa de limitar o livre acesso ao código implica na impossibilidade de distribuição do mesmo. Outras pessoas poderão utilizar a ferramenta criar suas próprias versões, porém, o portal oficial do BDDPM irá estimular que as contribuições sejam feitas de maneira organizada e centralizada através do repositório GIT da ferramenta.

---

[75] A GNU General Public License, GNU GPL, ou simplesmente GPL, é a designação da licença para software livre idealizada por Richard Matthew Stallman em 1989, no âmbito do projeto GNU da Free Software Foundation (FSF). GPL é a licença com maior utilização por parte de projetos de software livre. A GPL visa garantir que qualquer produto de software (regido por suas regras) possa ser compartilhado e modificado por qualquer pessoa e permaneça assim, indefinidamente. O software pode ser vendido, mas deve ser sempre aberto. A versão 2 da licença prega que qualquer tentativa de impedir esta liberdade acarreta na impossibilidade de distribuição do software modificado.

[76] A licença MIT, também chamada de licença X ou de licença X11, é uma licença de programas de computadores (software), criada pelo "Massachusetts Institute of Technology - MIT". Ela é uma licença que permite a reutilização de software licenciado em programas livres ou proprietários. Basicamente a única restrição imposta por ela é a de manter o aviso de copyright.

[77] http://www.fsf.org

[78] http://www.gnu.org/licenses/gpl-faq.en.html#WhatDoesCompatMean

[79] https://www.gnu.org/licenses/license-list.html#GPLCompatibleLicenses

Capítulo 4 – O BDDPM: Mais Detalhes

O BDDPM foi criado e testado em ambiente Windows com servidor WEB Apache[80] (httpd.apache.org), versão 2.2.26, com suporte a PHP (versão 5.5.34). Em teoria não existem impedimentos para que o sistema funcione em ambientes Linux com configuração semelhante, entretanto a solução não foi testada em tal cenário.

Um único Analista de Sistemas esteve envolvido no projeto de desenvolvimento do BDDPM: Alvaro Magnum Barbosa Neto, 30 anos, formado em Ciência da Computação pela Universidade Federal de Campina Grande – UFCG[81], especialista em Gestão de Projetos com PMBOK pela Faculdade Anglo Americano[82] e candidato à Mestre em Ciência da Computação pela UFPE[83]. O projeto foi acompanhado e supervisionado pelo Dr. Vinicius Cardoso Garcia[84], professor adjunto da UFPE e profissional com experiência em: Reengenharia de Software, Reuso de Software, Linhas de Produtos de Software, Arquitetura de Software, Programação Generativa, Computação na Nuvem, Datacenter as a Service, Backend as a Service e Social Machines.

A Seção 4.1 descreve a metodologia de desenvolvimento do plugin, trazendo os resultados de um questionário que foi aplicado no início do projeto para levantamento dos requisitos. Também descreve como foi seu desenvolvimento, arquitetura e testes utilizados. A 4.2 finaliza com uma demonstração prática de alguns dos recursos da ferramenta.

4.1. Metodologia de Desenvolvimento e Implantação do BDDPM

4.1.1. Fase 1: Levantamento de Requisitos

O início do desenvolvimento do BDDPM deu-se com a definição de seus requisitos. Para tal, foi utilizada uma simples técnica de investigação através da elaboração de um questionário com perguntas objetivas, e respostas de múltipla escolha, que foi respondido por 47 programadores / testadores de sistemas, 06 gerentes de desenvolvimento de software e 03 clientes de produto de software. O formulário pode ser conferido no Anexo 1. As justificativas e explanações para as perguntas são as seguintes:

[80] O servidor Apache (ou Servidor HTTP Apache, em inglês: Apache HTTP Server, ou simplesmente: Apache) é um popular servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications).
[81] http://www.ufcg.edu.br
[82] http://www.angloamericano.edu.br
[83] http://www.ufpe.br
[84] http://viniciusgarcia.com/about

- **PERGUNTA 1: Em sua opinião, para uma aplicação WEB ser considerada de fácil uso e simples, inclusive para usuários com pouco, ou nenhum, conhecimento técnico; qual deveria ser o número de telas úteis dessa aplicação? Considere que TELA ÚTIL é aquela que contém ou permite a entrada informações importantes e/ou essenciais para utilização do sistema.** A resposta para esta pergunta não se relaciona com o conhecimento da técnica do BDD por parte do interrogado. Baseia-se em sua opinião em relação a usabilidade e facilidade de uso considerando-se a quantidade de telas em um sistema. O BDDPM teria a quantidade de telas com maior quantidade de "votos" obtidos através do questionário.
- **PERGUNTA 2: Qual seu nível de experiência/conhecimento em relação ao BDD (Behavior Driven Development)?** Esta pergunta tem forte impacto nas demais respostas (exceto a pergunta 6), pois o entendimento do que é a técnica do BDD e a experiência com a mesma afetam o entendimento do que uma ferramenta voltada para o Behavior Driven Development faz, contém, ou deveria conter. Se a maior parte dos usuários tivesse respondido que desconheciam o BDD (letra a), então o questionário teria que ser aplicado novamente, dessa vez com a preocupação de seleção do público interrogado.
- **PERGUNTA 3: Em sua opinião, considerando uma aplicação WEB voltada para o BDD (Behavior Driven Development) e que permite a criação de testes de aceitação com Gherkin, qual a melhor proposta em termos de facilidade de uso? (Inclusive para usuários com pouco, ou nenhum, conhecimento técnico).** Esta foi, provavelmente, a pergunta mais difícil do questionário. Para respondê-la com propriedade seria necessário conhecer o BDD e o que é um teste de aceitação escrito em Gherkin, com toda sua estrutura e regras. O questionamento também envolve alguns conceitos de componentes de interface com o usuário (text box, combo box, radio, etc.). O interrogado teria que usar a imaginação para conceber mentalmente uma tela com o uso dos componentes indicados e objetivando a criação de testes de aceitação. Por um lado, estes pensamentos poderiam ser completamente distintos do que foi imaginado quando da elaboração da questão por parte do avaliador; por outro dava liberdade para que o avaliado refletisse e chegasse as suas próprias conclusões, que poderia também envolver a utilização dos mesmos componentes propostos nas opções de resposta. Assim sendo, ainda que a interface e/ou disposição dos elementos sejam imaginadas de maneira diferente por cada interrogado, o fator "escolha dos componentes" é quem teria o maior peso na hora de criar o BDDPM.
- **PERGUNTA 4: Na sua opinião, qual seria o recurso mais importante em uma aplicação WEB voltada para o BDD (Behavior Driven Development)?** O objetivo aqui seria entender o que um potencial usuário do BDDPM esperaria encontrar como principal funcionalidade, considerando a técnica do BDD. Caso a maior parte das respostas estivesse concentrada na letra E, isto é, que nenhuma das

funcionalidades listadas eram consideradas prioritárias, um novo questionário teria que ser aplicado, dessa vez incluindo a opção para especificar qual o recurso desejado. A versão utilizada do questionário não permitia a possibilidade de inclusão de respostas subjetivas/abertas em nenhum item.

- **PERGUNTA 5: Na sua opinião, qual seria o recurso mais importante PARA O CLIENTE DE UM PRODUTO DE SOFTWARE em uma aplicação WEB voltada para o BDD (Behavior Driven Development)?** A justificativa é a mesma da pergunta 4, aqui, entretanto, é sugerido que o questionado se coloque na posição de cliente.
- **PERGUNTA 6: Você é...** Para melhor avaliar as respostas das perguntas de 1 a 5, era importante conhecer o papel do avaliado, uma vez que o questionário foi enviado para uma quantidade indefinida de pessoas de diferentes empresas através de e-mail.

Os resultados obtidos, e suas respectivas avaliações, foram os seguintes:

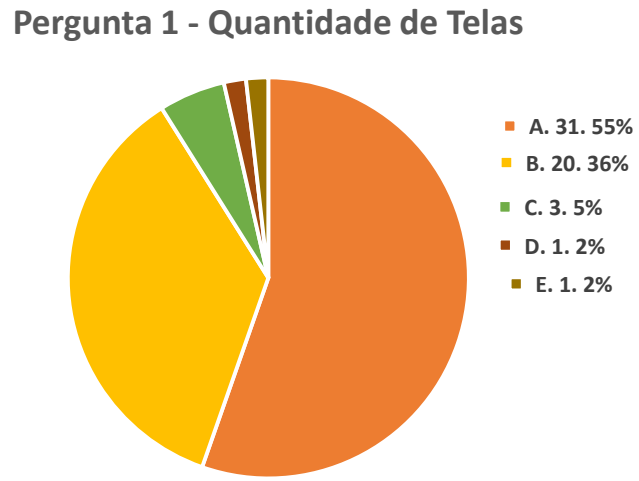

Figura 6 – Questionário para Interface. Resultado da Pergunta 1

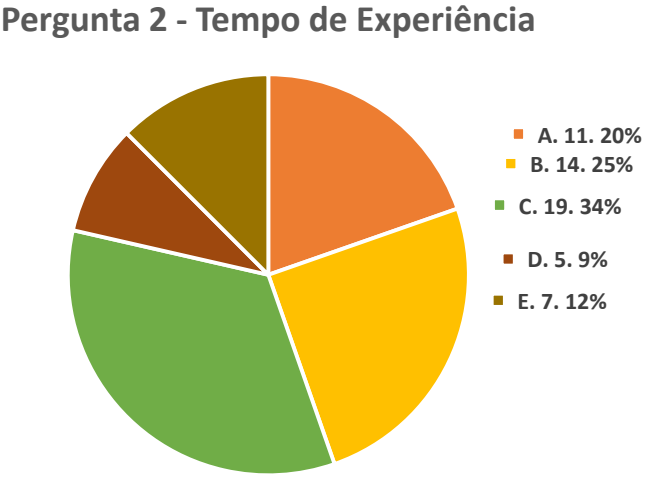

Figura 7 – Questionário para Interface. Resultado da Pergunta 2

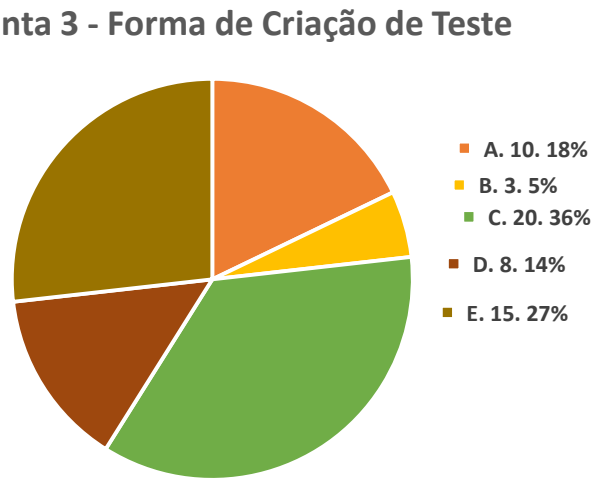

Figura 8 – Questionário para Interface. Resultado da Pergunta 3

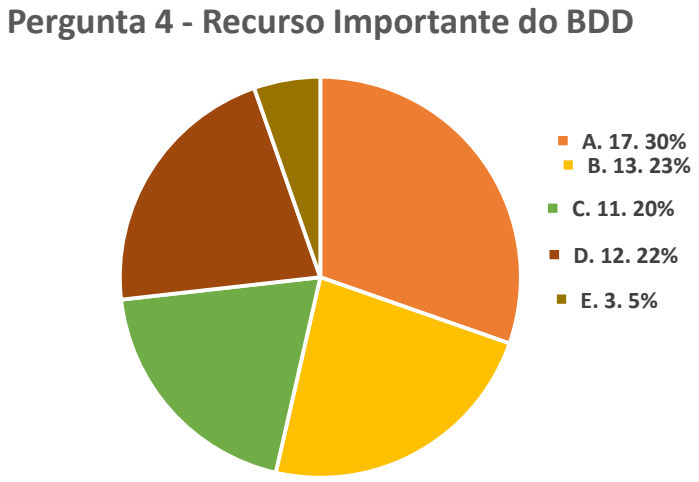

Figura 9 – Questionário para Interface. Resultado da Pergunta 4

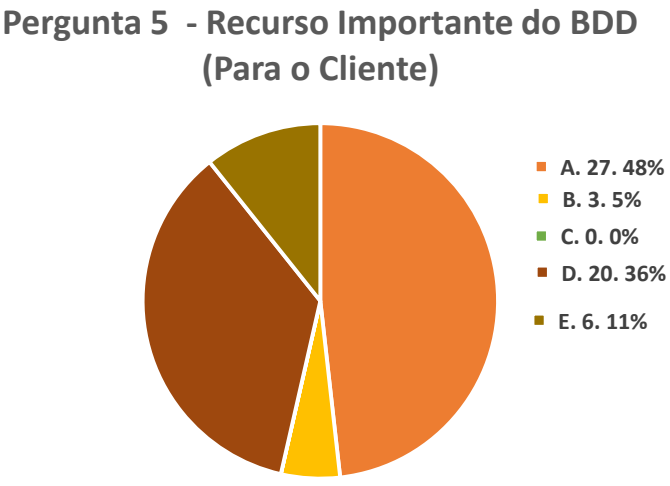

Figura 10 – Questionário para Interface. Resultado da Pergunta 5

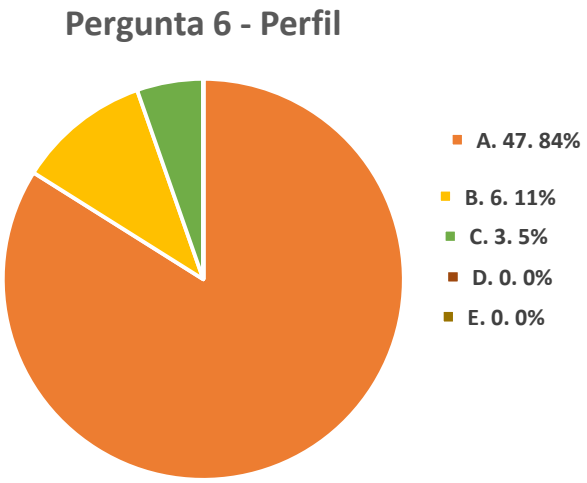

Figura 11 – Questionário para Interface. Resultado da Pergunta 6

Na primeira pergunta, a letra A (de 1 a 4 telas) foi a mais votada. Provavelmente pelo fato de que muitos usuários acreditam que, quanto menor a complexidade de algo, e quanto mais simples este algo for, melhor. Esta teoria é conhecida como o princípio KIS (Keep It Simple). Os avaliados provavelmente acreditam que uma quantidade menor de telas torna a utilização da aplicação mais fácil e simples. Para a segunda pergunta, a letra de maior marcação foi a letra C, indicando que 34% das pessoas avaliadas tinham menos de 01 ano de experiência com o BDD. É um grupo grande de pessoas. A terceira questão também teve a letra C com maior número de marcações. Esta reposta influenciou o desenvolvimento do BDDPM que utilizou os componentes citados (combo box e text box) em o padrão Wizard para montagem dos testes de aceitação. Para as perguntas 4 e 5, que são iguais e mudam apenas o ponto de vista de análise, todas as propostas sugeridas como respostas foram implementadas, exceto a funcionalidade de execução automática dos testes de aceitação. Finalmente, os resultados obtidos através da sexta pergunta possibilitaram a classificação dos avaliados.

Com base nos resultados obtidos através do questionário definiu-se que o BDDPM teria 04 telas principais:

- **Tela de Início** – Contém as informações de resumo referente ao projeto de desenvolvimento de software. Apresenta os quantitativos gerais de resultados testes (testes com falha, testes que passaram, testes que ainda não foram executados, etc.). Também apresenta um log de atividades dos usuários da ferramenta.
- **Tela de Configuração** – Concentra todas as configurações possíveis do plugin, incluindo os caminhos do sistema para o aplicativo GIT, a seleção do tipo de repositório (local ou remoto), as configurações de dados de acesso ao repositório (URL, login/usuário e senha) e os dados referente ao código do projeto em C#.
- **Tela de Casos de Uso** – É a tela que será utiliza pelos usuários para criação/adição dos testes de aceitação utilizando formulários com campos de múltipla escolha e campos de texto abertos organizados em um

Wizard. Na tela também é possível classificar os testes em níveis de prioridade (normal, importante, muito importante), além da edição e remoção dos testes.

- **Tela de Resultado de Testes** – Esta é a tela de acompanhamento em tempo real dos testes realizados com base nos testes de aceitação escritos pelos usuários. O acompanhamento é feito em tempo real: sempre que um teste é executado, o resultado é atualizado na interface com o usuário.

A interface do usuário foi pensada de maneira a facilitar/exibir/simplificar/otimizar os seguintes recursos: (i) **Possibilidade de acompanhamento do resultado dos testes**; (ii) **Geração de Código de Testes** e; (iii) **Escrita de Testes de Aceitação de forma Simplificada**. O sistema também dispõe de outras telas secundárias: (i) **sobre a ferramenta**; (ii) **licença da ferramenta**; (iii) **ajuda da ferramenta** e; (iv) **contato com os desenvolvedores**.

4.1.2. Fase 2: Desenvolvimento e Testes

A segunda etapa envolveu a criação da ferramenta propriamente dita, todo o processo de programação e junção dos componentes de terceiros utilizados. O processo de desenvolvimento fez uso das técnicas de TDD com Testes de Unidade através do Visual Studio e de BDD com o PHP Storm, Behat, Selenium Server e PHPUnit.

Um resumo, em números, dos quantitativos relacionados a testes podem ser conferidos na Tabela 5:

Descrição	Quantidade
Testes de Comportamento	21
Testes de Unidade	83
Arquivos Gherkin	18
Classes de Teste	19

Tabela 5 – Testes do BDDPM em números

Todos os 83 testes de unidade foram executados a partir de uma única classe. Já os 21 testes de comportamento foram distribuídos em 18 classes de testes, cada uma responsável por tratar os testes de aceitação descritos nos 18 arquivos Gherkin. Ainda em relação os testes, foi utilizado o padrão Facade/Fachada[85] para centralizar as chamadas dos testes em um único ponto central (fachada). Sendo assim, embora existam 19 classes de testes que separam os testes de maneira lógica, classificados de acordo com o que é testado, todos eles chamam uma única classe de acesso (fachada) para execução dos testes. A arquitetura dessa estratégia pode ser vista na Figura 12:

[85] O padrão de projeto Facade é um padrão estrutural que fornece uma interface unificada para um conjunto de interfaces em um subsistema. O padrão Facade define uma interface de nível mais elevado que faz com que o subsistema fique mais fácil de ser utilizado. De forma mais simples podemos dizer que o padrão Facade tem como objetivo esconder a complexidade de um sistema expondo apenas as interfaces que o cliente precisa enxergar. Com isso o sistema fica mais simples e fornece uma coleção de métodos mais fáceis de entender.

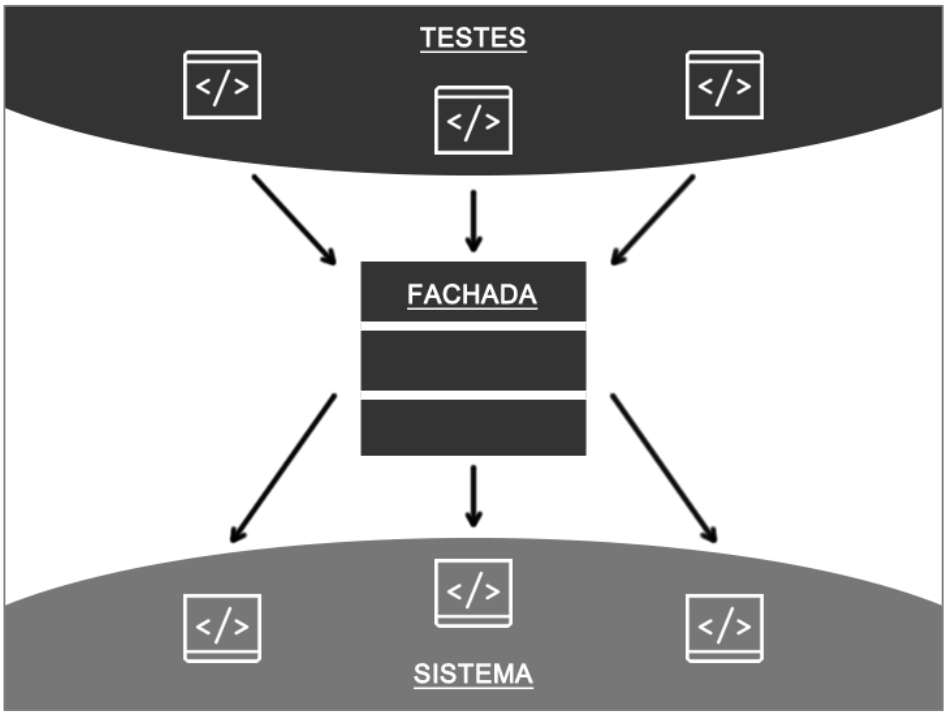

Figura 12 – Arquitetura de Testes do BDDPM

A lista dos requisitos esperados na definição inicial do BDDPM era a seguinte:

1. O plugin deve ser de rápida instalação
2. O plugin deve ser de rápida desinstalação
3. Deve ser uma ferramenta de página única.
4. A ferramenta será acessível por três papéis básicos: gestor do projeto de software, testador/desenvolvedor do projeto de software e cliente do produto de software.
5. A partir da página inicial deve ser possível ter acesso ao log do sistema na página inicial
6. A partir da página inicial deve ser possível ter acesso ao resumo dos testes na página inicial
7. A ferramenta deve ter integração com o GIT (O GIT deve estar instalado na máquina servidora)
8. A ferramenta deve permitir a configuração do path (caminho) do sistema para o GIT, além da configuração de login/usuário, URL e senha do repositório.
9. A ferramenta deve suportar repositórios GIT remotos e locais
10. O plugin deve permitir algumas configurações do projeto de software (nome do pacote, diretórios utilizados para salvar os arquivos, nomenclatura de classes, etc.)
11. A ferramenta deve possibilitar a criação, edição e remoção de cenários de testes de aceitação
12. A ferramenta deve possibilitar a criação, edição e remoção de passos dos cenários de testes de aceitação.
13. A ferramenta deve prover um meio de pesquisa de testes de aceitação por conteúdo.

14. Os testes de aceitação deverão poder ser classificados em: normal, importante e muito importante.
15. A classificação dos testes de aceitação deverá seguir um padrão de cores: cinza, amarelo e vermelho para as classificações de normal, importante e muito importante, respectivamente.
16. O plugin deve permitir a construção/criação de testes de aceitação sem o conhecimento da estrutura/sintaxe do Gherkin
17. O Gherkin deve ser gerado automaticamente a partir das entradas do usuário.
18. Deve ser possível o envio e sincronização automática dos arquivos de testes de aceitação escritos em Gherkin para/com o repositório do projeto.
19. O código de teste de software em C# deve ser gerado automaticamente pela ferramenta.
20. O código de testes de software em C# deve ser de maneira tal que, a cada execução, se comunique com a ferramenta para enviar o resultado do teste executado.
21. Deve ser possível o envio e sincronização automática dos arquivos de códigos de testes escritos em C# para/com o repositório do projeto.
22. O cliente, ao utilizar a ferramenta, deverá poder ver os resultados dos testes em tempo real (sempre que forem executados). Cada teste pode ter o um dos seguintes status: sucesso, falha ou inconclusão.
23. Os status de erro deverão seguir um padrão de cores: verde, vermelho e amarelo para sucesso, falha e inconclusão, respectivamente.
24. Deverá ser possível acessar a página de ajuda da ferramenta através da interface com o usuário em todas as telas do sistema.
25. Deverá ser possível acessar a página com as informações do sistema através da interface com o usuário em todas as telas do sistema.
26. Deverá ser possível acessar a página que descreve a licença do plugin através da interface com o usuário em todas as telas do sistema.
27. Deverá ser possível acessar a página de contato com a equipe desenvolvedora do sistema através da interface com o usuário em todas as telas do sistema.
28. Os resultados dos testes devem ser atualizados em tempo real (sempre que cada teste for executado) na interface do usuário.

Estes requisitos foram transformados em testes de comportamento e foram criados os testes de aceitação descritos em Gherkin para utilização do BDD. Além disso, as funções e algumas unidades de código eram testadas através de testes de unidade.

Com todos os requisitos em mãos e com os testes criados, o desenvolvimento do BDDPM durou 90 (noventa) dias. Na figura 23 pode ser conferida a arquitetura da versão final do projeto, que será chamada de Arquitetura do Sistema BDDPM[86]:

[86] Sistema BDDPM – Para fins deste trabalho, será considerado Sistema BDDPM o conjunto de todos os componentes da Arquitetura do Projeto BDDPM.

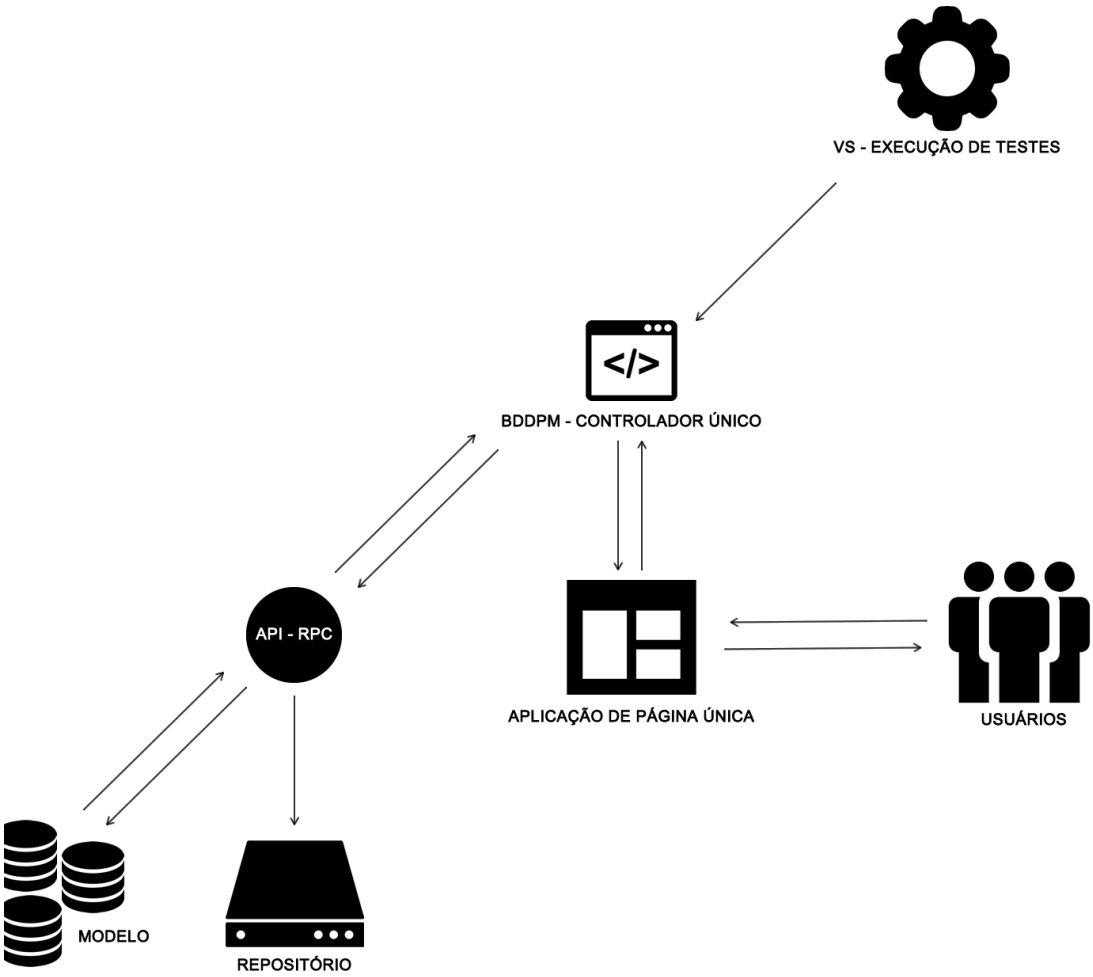

Figura 13 – Arquitetura do Sistema BDDPM

O BDDPM possui um único ponto de acesso, o “Controlador Único”. Todos os clientes que quiserem fazer uso dos recursos da ferramenta devem acessar esta camada. A priori existem dois tipos de clientes: os usuários humanos (clientes, desenvolveres, testadores e gerentes de projetos de software); e os usuários não humanos. Outros clientes podem se beneficiar das funcionalidades do BDDPM uma vez que o plugin provê uma interface de comunicação que segue os padrões do RPC[87]. Sendo assim, conhecendo-se a API do BDDPM pode-se fazer chamadas remotas aos seus métodos e funções a partir de quaisquer outros sistemas.

Os usuários acessam o controlador para ter acesso a interface WEB do plugin. Eles interagem com esta visão enviando dados e observando os resultados de maneira visual. O VS (Visual Studio), por sua vez, se comunica com o BDDPM apenas enviando dados: os resultados dos testes. Toda vez que um teste é executado, o VS envia as informações contendo o retorno da avaliação (sucesso, falha ou inconclusivo)

[87] Em Ciência da Computação, a Chamada Remota de Procedimento (RPC, acrônimo de Remote Procedure Call) é um princípio/protocolo de comunicação que permite que programas de computador possam executar procedimentos em outros sistemas computacionais, ainda que eles estejam em diferentes domínios em uma rede. Esta conexão se dá de maneira tal que seus detalhes de implementação são transparentes: do ponto de vista do código, a chamada se assemelha à de métodos locais; só que estes procedimentos são executados remotamente em outro ambiente. Muito utilizado para implementação do modelo cliente-servidor de computação distribuída.

para a ferramenta através de uma requisição HTTP[88], pois o plugin roda em cima de uma aplicação WEB.

Uma API[89] (API – RPC) permite o acesso a todos os recursos públicos do BDDPM (adicionar teste, remover teste, salvar configurações, obter resultados de testes, enviar teste para o repositório, etc.). É neste módulo que está concentrada a lógica de negócio (do lado do servidor) do BDDPM responsável pela recepção, manipulação, transformação e repasse de dados para a camada da lógica de domínio da aplicação: a camada de Modelo. Esta API, que foi desenvolvida em PHP, também é capaz de se comunicar com o repositório do projeto de software através da criação e envio de arquivos para o mesmo, porém não recebe nenhuma informação deste.

Dependendo de onde veio a requisição, o "Controlador Único" do BDDPM faz a chamada da lógica de negócio correta através da API. Quando o cliente do controlador é o VS, há uma comunicação do tipo simplex[90], onde o transmissor é o Visual Studio e o receptor é o Controlador. Esta comunicação (VS → Controlador) juntamente com a comunicação da API com o Repositório da aplicação testada (em GIT) são as únicas comunicações simplex do sistema, as demais comunicações são do tipo duplex[91].

O Controlador foi desenvolvido em PHP seguindo as definições constantes no manual do desenvolvedor[92] do Mantis, que estabelece uma arquitetura voltada para eventos: ações, comandos e interações com o usuário tornam-se eventos que são capturados pela API que executa as ações correspondentes no servidor.

Na camada da lógica de domínio (o Modelo), os dados são persistidos e passam a ter sentido: entradas dos clientes tornam-se testes, resultado de testes são classificados, etc. O dado, que é algo quantificável e que pode ser guardado, transforma-se em informação persistida, que é algo que tem significado e transmite mensagem. Este é um módulo PHP com persistência em Banco de Dados MySQL através da técnica de ORM.

---

[88] HTTP é sigla de HyperText Transfer Protocol que em português significa "Protocolo de Transferência de Hipertexto". É um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores, principalmente na Internet. O HTTP funciona como um protocolo de requisição-resposta no modelo computacional cliente-servidor. Um navegador WEB, por exemplo, pode ser o cliente e uma aplicação em um computador que hospeda um site da web pode ser o servidor. O cliente submete uma mensagem de requisição HTTP para o servidor. O servidor, que fornece os recursos, como arquivos HTML e outros conteúdos, ou realiza outras funções de interesse do cliente, retorna uma mensagem resposta para o cliente. A resposta contém informações de estado completas sobre a requisição juntamente com o conteúdo solicitado pelo cliente.

[89] API, de Application Programming Interface (em português: Interface de Programação de Aplicações) é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

[90] Uma comunicação é dita simplex quando há um dispositivo emissor e outro dispositivo receptor, sendo que este papel não se inverte no período de transmissão. A transmissão tem sentido unidirecional, não havendo retorno do receptor.

[91] Duplex é um sistema de comunicação composto por dois interlocutores que podem comunicar entre si em ambas direções. Diz-se, portanto, bidirecional.

[92] http://www.mantisbt.org/docs/master-1.2.x/en/developers/dev.plugins.building.html

Por fim, a “Aplicação de Página Única” é uma parte do Sistema BDDPM que tem sua própria arquitetura conforme exibida na Figura 14:

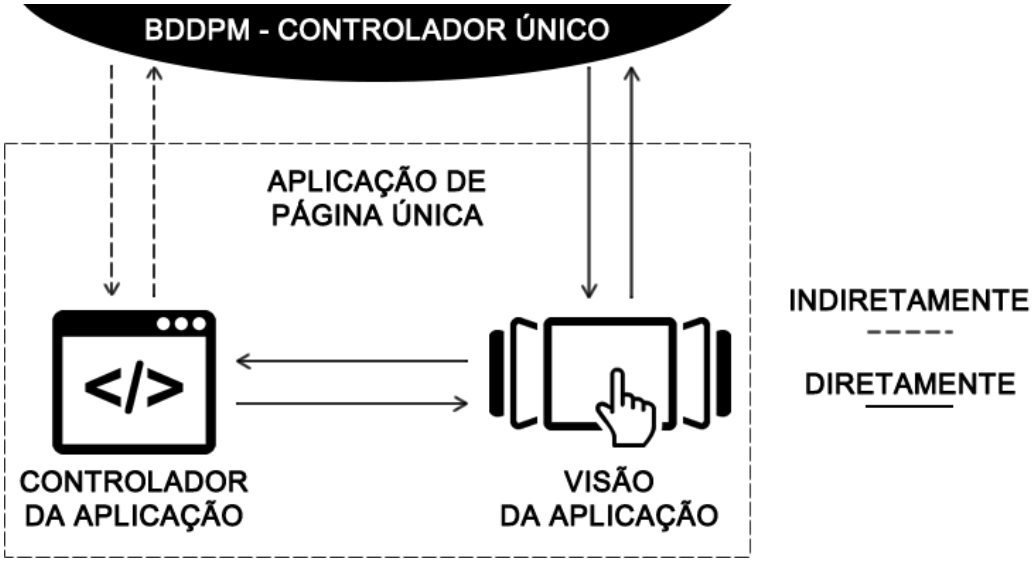

Figura 14 – Arquitetura da Aplicação de Página Única - BDDPM

Os principais recursos do BDDPM foram incluídos na “Aplicação de Página Única” que executa no lado do cliente. O “Controlador Único” do BDDPM, a API e os componentes que a ela se relacionam rodam no lado do servidor. Não há a camada de Modelo no cliente, uma vez que ela é acessível através da comunicação entre o “Controlador da Aplicação” e o “Controlador Único” do BDDPM. A “Aplicação de Página Única” é formada por apenas 02 módulos: o “Controlador da Aplicação” e a “Visão da Aplicação”.

O “Controlador da Aplicação” contém a parte da lógica de negócio (do lado do cliente) do BDDPM relacionada a técnica do Behavior Driven Development e é responsável por processar todas as informações provenientes da interface com o usuário repassando-as para o servidor quando necessário. Também tem a tarefa de atualizar a interface WEB com o usuário de acordo com a situação. Este módulo se comunica indiretamente com o “Controlador Único”, pois depende de requisição da interface (Visão da Aplicação).

A “Visão da Aplicação” é um conjunto de telas que são trocadas dinamicamente de acordo com as interações dos usuários. Embora existam diversas páginas distribuídas em arquivos HTML, logicamente elas são exibidas e carregadas como se fossem uma única página para o usuário. Enquanto a lógica da aplicação fica dividida e modularizada, para o cliente este processo é transparente e ele percebe apenas uma página funcionando.

Tanto o “Controlador da Aplicação” como a “Visão da Aplicação” utilizam JS (JavaScript) de maneira que as principais funcionalidades do BDDPM sejam independentes de linguagem de servidor; isto é, o BDDPM só depende do servidor para realização de duas principais atividades: (i) persistência de informações e (ii) acesso ao repositório do projeto do software que está sendo testado. As outras funcionalidades (relacionadas à técnica do BDD) rodam no navegador/browser. Esta

estratégia facilita a migração do plugin para outras plataformas WEB de gestão de projetos em outras linguagens (Java, C#, Python, Ruby, etc.).

A junção destas arquiteturas, formando o Sistema BDDPM, obedece ao padrão MVC[93]. O Modelo é representado pela própria camada de Modelo, a Visão é fornecida pela camada da "Aplicação de Página Única"; e a "API – RPC" e o Controlador Único do BDDPM são responsáveis pela parte do Controlador.

4.1.3. Fase 3: Implantação

A última etapa de desenvolvimento do BDDPM envolveu sua instalação em um ambiente de produção real para avaliação de resultados e impactos da ferramenta como proposta de solução para o problema.

A empresa escolhida para implantação e utilização do BDDPM foi a PBprev (Paraíba Previdência). De acordo com o próprio portal (http://www.pbprev.pb.gov.br) da instituição, ela se intitula como uma "*autarquia criada pela Lei Estadual nº 7.517, de 30 de dezembro de 2003. Por força do art. 7º da Lei nº 7.721/2005, a PBPREV encontra-se vinculada à Secretaria de Estado do Governo da Paraíba*".

Compete ao órgão gerir o regime próprio de previdência dos servidores públicos efetivos do Estado da Paraíba de acordo com os princípios jurídicos e legais estabelecidos na Constituição brasileira e por força de lei.

O critério utilizado para escolha foi o acadêmico: este trabalho é uma dissertação de Mestrado Profissional (MPROF) em Ciência da Computação oferecido pelo Centro de Informática da UFPE (Universidade Federal de Pernambuco) que em seu site prega o seguinte[94]: "*O MPROF difere do Mestrado Acadêmico por* ***lidar com problemas que o profissional vivencia dentro do mercado****. Sendo assim,* ***os projetos de conclusão normalmente tratam de temas de interesse direto da empresa em que o mestrando trabalha****. A interação dos acadêmicos com as problemáticas enfrentadas no dia-a-dia das instituições* ***contribui para a melhoria das empresas****, e consequentemente para o desenvolvimento da região*". A empresa no qual o mestrando trabalha (trecho sublinhado) é a PBprev.

O BDDPM foi instalado na instituição com autorização do Gerente de Tecnologia da empresa em dezembro de 2014. O plugin foi utilizado no projeto de desenvolvimento do projeto "SISPROTO – SAC" (SSAC). O SSAC (lê-se SAC, primeiro "S" mudo), é um pequeno[95] sistema de 02 módulos, um desktop e outro WEB, que permite os funcionários da instituição solicitem correções nos processos que tramitam dentro da empresa. Nesse caso, o processo é um documento eletrônico que é criado quando da entrada de uma aposentaria ou pensão de um servidor público

[93] Model-View-Controller (MVC), em português Modelo-Visão-Controlador, é um padrão de projeto de software que separa a informação (e as suas regras de negócio) da interface com a qual o usuário interage. É uma forma de estruturar uma aplicação de forma que sua interface (View) esteja separada do controle da informação que ela manipula (Model), separação essa que é intermediada por uma camada controladora (Controller).

[94] http://www2.cin.ufpe.br/site/secao.php?s=3&c=116

[95] Pequeno, nesse casso, no sentido de ser uma ferramenta de baixa complexidade e com projeto de desenvolvimento de curto prazo.

estadual. Estes documentos podem conter dados errados: o telefone de contato de um servidor, por exemplo. O módulo WEB do SSAC fornece a interface para que as correções/alterações sejam solicitadas, e o módulo Desktop funciona como um serviço que fica monitorando todas as solicitações feitas através da interface WEB. Sempre que uma solicitação for detectada, o programa notifica um funcionário de TI da Paraíba Previdência para que ele possa fazer o atendimento da solicitação.

Após o período de 45 dias de desenvolvimento do projeto o BDDPM ainda foi avaliado por mais 30 (trinta) dias através da técnica Goal-Question-Metric (explanada em detalhes no próximo capítulo). Juntando-se os 02 períodos, o BDDPM passou por 75 dias de avaliação. Durante este período a ferramenta foi avaliada sob diferentes perspectivas:

- Validação do BDDPM – O BDDPM cumpriu com o que foi prometido?
- Falhas – Problemas e bugs no software do BDDPM
- Vantagens – Que valores foram agregados ao projeto devido a utilização do BDDPM.
- Adequação ao BDD – O BDDPM realmente permite a fácil utilização do BDD no projeto de desenvolvimento de software?
- Desempenho da Ferramenta – A ferramenta BDDPM é eficiente?
- Desempenho do projeto no qual o BDDPM foi utilizado – O projeto foi mais eficiente com o uso do BDDPM?
- Validação do produto de software do projeto no qual o BDDPM foi utilizado – Foi criado o produto correto?
- Verificação do produto de software do projeto no qual o BDDPM foi utilizado – O produto foi criado corretamente?

4.2. Recursos do BDDPM na Prática

Nesta Seção os recursos do BDDPM serão descritos **através da exemplificação de sua utilização em um projeto real**: o projeto SSAC, ou seja, ao mesmo tempo em que é mostrado como a ferramenta é utilizada, também são destacados os seus recursos.

4.2.1. Configuração

A Configuração do BDDPM é dividida em duas partes: a “Configuração do Plugin” e a “Configuração do Projeto”. A “Configuração do Plugin” contém opções relacionadas à própria ferramenta do BDDPM. As duas partes são tratadas de maneira separada, de maneira que é preciso salvar cada uma individualmente, sempre que forem alteradas. Atualmente, a única opção funcional é a de configuração do caminho para o executável do GIT. É necessário ter o GIT instalado no ambiente de produção para que o BDDPM funcione.

É importante destacar que, sem este recurso instalado, não será possível a utilização do BDDPM. Eis os passos para fazer a configuração adequada da ferramenta (a configuração é feita pelo gerente do projeto):

1. Ao acessar o BDDPM, selecionar a opção “Configuration”:

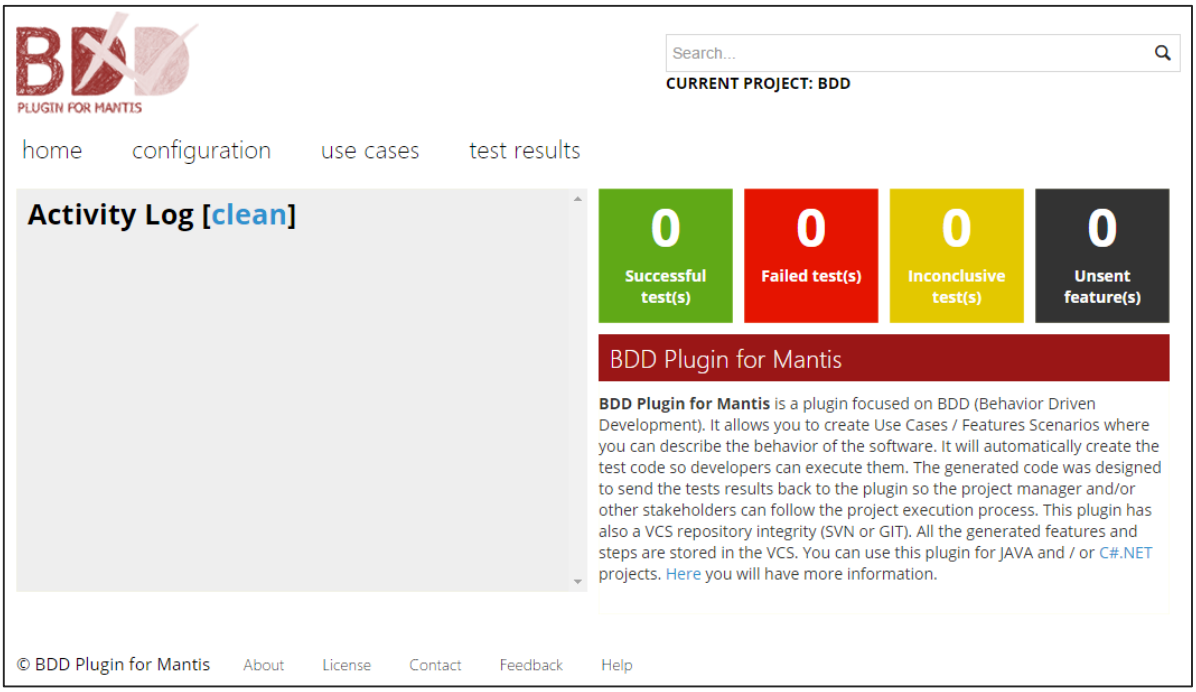

Figura 15 – Tela inicial do BDDPM após a instalação

2. A primeira parte da configuração envolve a “Configuração do Plugin”, onde é possível configurar o caminho para o GIT no ambiente de produção. Este caminho deve ser completo, incluindo o próprio executável. A versão do git utilizada no projeto foi a 1.9.4 de junho de 2014. O caminho de instalação do git foi o “C:\Program Files (x86)\Git\bin\git.exe”. Após a inclusão das informações, elas foram salvas através do botão “Save Config”:

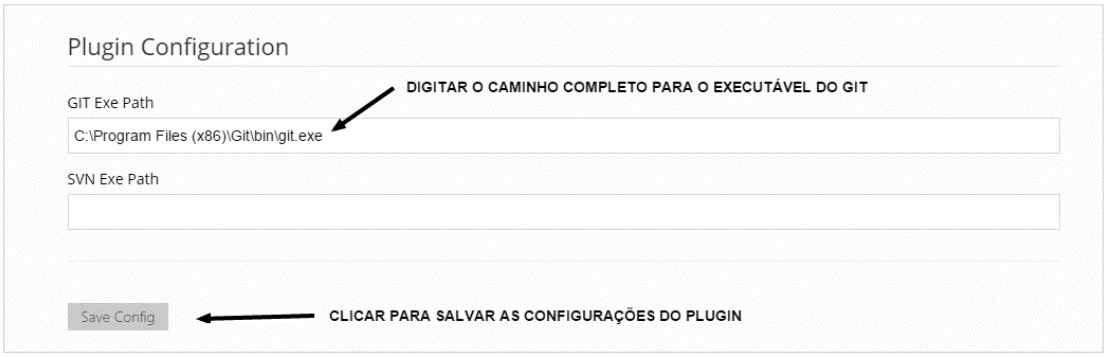

Figura 16 – BDDPM: Configuração do Plugin

3. A segunda e última parte da configuração envolve a “Configuração do Projeto”. O projeto se refere ao projeto de desenvolvimento de software que está sendo testado. O BDDPM sempre altera as configurações do projeto corrente (projeto selecionado no Mantis). Esta etapa possui oito configurações: (i) a seleção do tipo de repositório/VCS. Apenas o GIT está disponível; (ii) o tipo do repositório – remoto/online ou local; (iii) o caminho completo para o repositório do projeto, seja ele local ou remoto; (iv) o login do usuário utilizado para acessar o repositório; (v) a senha do usuário utilizada para acessar o repositório; (vi) o

namespace[96]/pacote do projeto de testes; (vii) o diretório, a partir da raiz do repositório, onde serão salvos os arquivos com os testes de aceitação descritos em gherkin; e (viii) o diretório, a partir da raiz do repositório, onde serão salvos os arquivos com os códigos de testes em C#. O projeto do SSAC foi colocado em um repositório GIT local, sem necessidade de login e senha, acessível através do caminho “C:\dev\SSAS\SSAS.Web”. O namespace do projeto de testes foi o “SASS.Tests”, o diretório onde os testes de aceitação foram salvos foi o “SASS.Tests\Features” e o diretório onde os códigos de testes foram salvos foi o “SASS.Tests\Steps”[97]. Após a inclusão das informações, elas foram salvas através do botão “Save Config”:

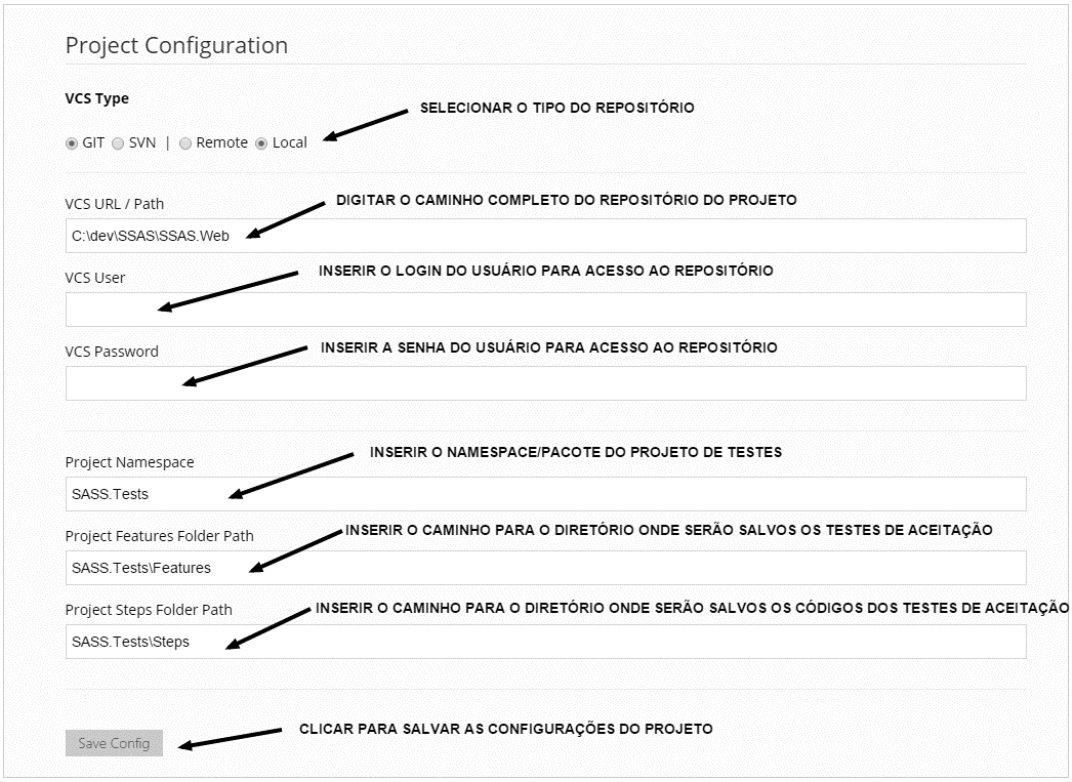

Figura 17 – BDDPM: Configuração do Projeto

4.2.2. Criação dos Testes de Aceitação

Ao todo, no projeto do SSAC, foram gerados 56 (cinquenta e seis) testes de comportamento distribuídos em 43 arquivos de testes de aceitação escritos em Gherkin. Não é objetivo deste trabalho detalhar o processo de desenvolvimento do SSAC. Além disso, a exposição de detalhes de código e estratégias da solução não foi liberada pela presidência/diretoria da PBprev. Por este motivo, o recurso de criação

[96] Namespaces são usados intensamente nos programas de C# de duas maneiras. O principal objetivo é o de organizar as diversas classes que podem existir em um sistema, além de ajudar a controlar o escopo dessas classes. Possuem um conceito semelhante ao "Pacote/Package" do Java.

[97] Curiosidade: O erro de nomenclatura (de SSAC para SSAS) na configuração do projeto só foi percebido ao final do desenvolvimento do SSAC. Os caminhos foram utilizados com a nomenclatura incorreta até o final do projeto. O problema não impactou nos resultados e não criou impedimentos que pudessem comprometer o desenvolvimento do projeto ou quaisquer uma de suas funcionalidades.

de testes do BDDPM será demonstrado através de UM ÚNICO exemplo do SSAC, cujo comportamento foi obtido através do seguinte requisito:

- A cada 5 minutos, o sistema desktop deverá checar se há novas solicitações de correções de processos. Caso exista alguma solicitação, deverá ser exibida uma notificação na tela do usuário Desktop (com opções de atendimento e recusa de atendimento), porém ela não deverá ser exibida caso algum funcionário de TI já a tiver atendido.

No BDDPM os comportamentos são descritos através de “Casos de Uso”, ou seja, em uma determinada situação de uso do software (caso de uso), espera-se um ou mais comportamentos. Ao final do processo, cada “Caso de Uso” criado no BDDPM irá gerar um arquivo escrito em Gherkin. Este processo é transparente para o usuário, ou seja, ele (cliente) cria o teste em alto nível e, em background (segundo plano), o plugin irá fazer a conversão do teste para a linguagem. Cada “Caso de Uso” também descreve uma Funcionalidade/Feature que contém um ou mais cenários de utilização da aplicação. Para todos os fins deste trabalho, os termos Caso de Uso, Funcionalidade e Feature serão utilizados como sinônimos e como conteúdos de um teste de aceitação.

Diferentemente das opções de configuração do BDDPM, que devem ser modificadas pelo gestor do projeto, os recursos de criação de testes de aceitação podem, e **deveriam**, ser utilizados pelo cliente do projeto. Com base no requisito citado anteriormente, o **cliente** do SSAC (o setor de TI representado por seu coordenador) criou o seguinte “Caso de Uso” (a numeração das instruções foi inserida para fins didáticos):

Funcionalidade: *Notificações*
Como um desenvolvedor e usuário da versão Desktop do SSAC, o sistema deverá monitorar novas solicitações a cada 5 minutos e exibir notificações sempre que encontrar uma nova solicitação.

Cenário: *Notificação de Nova Solicitação*
1 - Dado que o SSAC Desktop esteja aberto
2 - Dado que a cada 5 minutos o SSAC deve checar se existem novas solicitações
3 - Quando uma nova solicitação for identificada
4 - Então o SSAC deve exibir uma mensagem de alerta a respeito da solicitação
5 - E o alerta deve conter as opções de atendimento e de recusa de atendimento da solicitação
6 - Mas o alerta não deverá ser exibido caso a notificação já esteja marcada com em atendimento

Figura 18 – Requisito do SSAC transformado em teste de comportamento

Este teste de comportamento pode ser criado no BDDPM seguindo-se os seguintes passos:

1. Selecionar a opção “Use Cases” para ter acesso a tela de criação dos testes de aceitação:

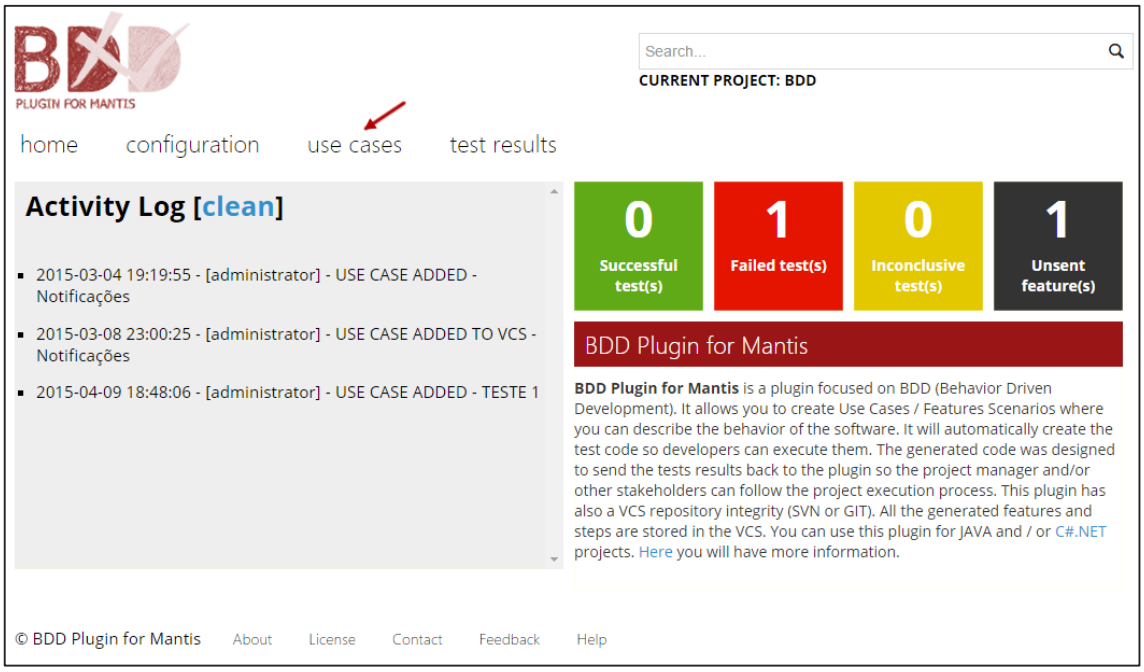

Figura 19 – BDDPM: Opção de acesso à página de criação de testes

2. O primeiro conjunto de opções para criação de um teste de aceitação engloba a inserção do (i) título da funcionalidade/feature e (ii) e da descrição da funcionalidade/feature. No caso em específico foram inseridos os seguintes conteúdos "*Notificações*" e "*Como um desenvolvedor e usuário da versão Desktop do SSAC, o sistema deverá monitorar novas solicitações a cada cinco minutos e exibir notificações sempre que encontrar uma nova solicitação.*", respectivamente:

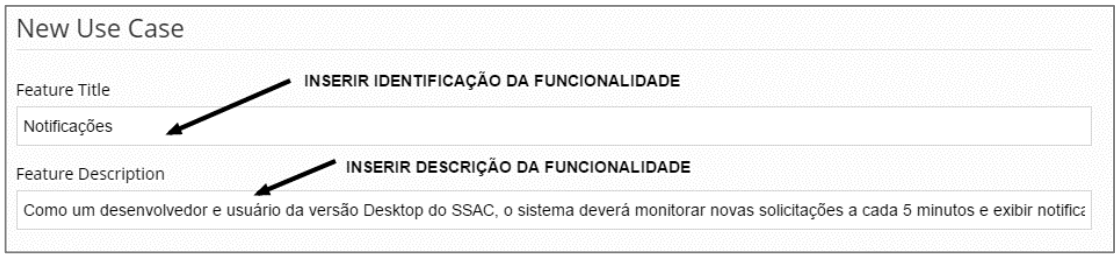

Figura 20 – BDDPM: Criação de Caso de Uso (funcionalidade e descrição)

3. Em seguida, deve-se escolher a prioridade do teste que será executado. Existem três opções: normal, importante e muito importante; sendo que a prioridade "Normal" é a de menor relevância e a prioridade "Muito Importante" é a de maior relevância. Além da classificação por nomenclatura e significado, também há a classificação por cores: cinza, amarelo e vermelho, respectivamente. A prioridade do teste será utilizada por desenvolvedores e testadores na hora da criação e execução dos testes, sendo que as funcionalidades de maior prioridade receberão maior atenção. Também é um recurso útil para clientes e gerentes do projeto para que eles possam acompanhar quais comportamentos prioritários já foram implantados e

testados. A funcionalidade “Notificações” foi classificada com maior prioridade (Muito Importante) no projeto do SSAC.

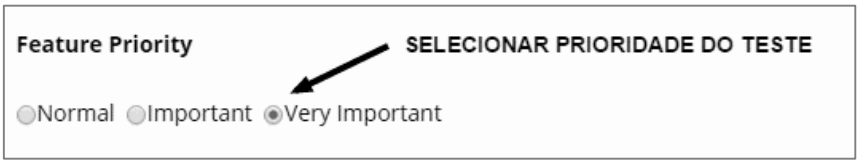

Figura 21 – BDDPM: Criação de Caso de Uso (prioridade do teste)

4. Na sequência é possível escolher a linguagem de programação utilizada para geração dos códigos dos testes. Atualmente a opção funcional é a linguagem C#, linguagem do projeto SSAC:

Figura 22 – BDDPM: Criação de Caso de Uso (linguagem dos testes)

5. Na parte final é disponibilizado um formulário seguindo o padrão Wizard, onde é possível inserir instruções passo a passo. O BDDPM segue a estrutura de um documento escrito em Gherkin. Cada cenário de utilização de um software tem uma descrição resumida do comportamento e um conjunto de instruções que descreve este comportamento de forma sequencial. O cliente não precisa se preocupar com a estrutura do Gherkin, nem com a ordem dos elementos, pois o BDDPM oferece as opções na sequência correta. O cliente precisa apenas escolher os tipos de instruções (dado que, quando, então, etc.) e preencher os campos. Existem botões de adição e remoção de cenários e de adição e remoção de instruções. Eles estão o tempo todo visíveis no formulário. O caso de uso descrito pelo cliente do SSAC envolve 06 instruções (ver Figura 34). As instruções foram inseridas conforme a Figura 23:

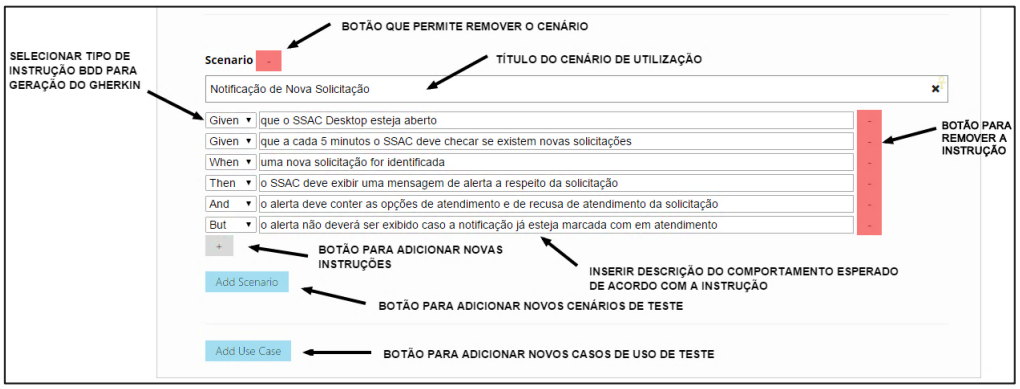

Figura 23 – BDDPM: Criação de Caso de Uso (descrição do comportamento)

É importante perceber que, no BDDPM, o cliente do produto de software não precisa entender sobre a estrutura e conceitos do Gherkin. De fato, ele não precisa nem saber que o Gherkin existe, bastando apenas entender os tipos de instruções

(Dado que, Quando, Então, etc.) para descrever o comportamento esperado do software.

O repositório oficial do Cucumber descreve a linguagem Gherkin no seguinte endereço: http://goo.gl/1psVj. Ela é considerada uma "Linguagem Específica de Domínio"[98]. Um detalhe importante é que ela também é considerada uma linguagem "Business Readable", ou seja, que foca na possibilidade de leitura, no entendimento pelo cliente. É um conceito diferente do "Business Writable", cujo foco permite ao cliente, não apenas entender, mas também escrever na linguagem. Em seu site[99], Martin Fowler entende que, embora uma linguagem "Business Writable" tenha grandes benefícios (inclusive com reconhecimento por parte dos clientes), alcançar algo nesse sentido pode ser bastante custoso.

O BDDPM procura dar um passo nesta direção sem especificar uma nova linguagem, mas passando a responsabilidade de formatação, estruturação e formalização (da mesma) para a máquina; deixando com o ser humano apenas a parte mais lógica, simples e intuitiva do processo de criação.

Esse mecanismo estimula a participação do cliente, não "apenas" como um auxiliar na escrita de testes, mas como o principal agente criador dos testes; tudo isso através da redução da "Curva de Aprendizado"[100] para adoção do BDD. Como resultado da criação do "Caso de Uso" via BDDPM, a ferramenta se encarrega de gerar o Gherkin conforme a Figura 24:

Feature: Notificações
Como um desenvolvedor e usuário da versão Desktop do SSAC, o sistema deverá monitorar novas solicitações a cada 5 minutos e exibir notificações sempre que encontrar uma nova solicitação.

@2
Scenario: Notificação de Nova Solicitação
Given que o SSAC Desktop esteja aberto
Given que a cada 5 minutos o SSAC deve checar se existem novas solicitações
When uma nova solicitação for identificada
Then o SSAC deve exibir uma mensagem de alerta a respeito da solicitação
And o alerta deve conter as opções de atendimento e de recusa de atendimento da solicitação
But o alerta não deverá ser exibido caso a notificação já esteja marcada com em atendimento

Figura 24 – BDDPM: Caso de Uso transformado em gherkin

A Figura 18 e 24 se assemelham bastante. Logicamente são a mesma coisa. A diferença é que o conteúdo da primeira foi escrito por alguém com conhecimento em BDD e Gherkin, e a segunda pode ser gerada por uma pessoa sem esses

[98] Em desenvolvimento de software, uma linguagem de domínio específico, do inglês: Domain-Specific Language (DSL), é um tipo de linguagem de programação, ou linguagem de especificação, para resolver problemas específicos, focadas em um domínio de problema particular.
[99] http://www.martinfowler.com/bliki/BusinessReadableDSL.html
[100] Curva de aprendizagem é uma representação do nível médio cognitivo de aprendizagem para uma determinada atividade ou ferramenta.

conhecimentos (através do BDDPM). O BDDPM também gera, automaticamente, os códigos de testes em C# conforme exemplo da a Figura 25:

```
    …

    [Given(@"que o SSAC Desktop esteja aberto")]
    public void GivenQueOSsacDesktopEstejaAberto()
    {
      ScenarioContext.Current.Pending();
    }

    ...

    #region BddPluginComunication

    // ------------------------------------------------------------------------
    //     FROM THIS POINT: This code was generated by BDD Plugin for Mantis 1.0
    //
    //     Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
    // ------------------------------------------------------------------------

    [AfterScenario]
    public void AfterScenario()
    {

      var error = ScenarioContext.Current.TestError;
      var id = ScenarioContext.Current.ScenarioInfo.Tags[0];

      …

      if (error == null)
      {
        status = "1";
      }
      else
      {
        errorMessage = Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(error.Message));
        status = "0";
      }

      WebRequest.Create(string.Format(url, id, status, errorMessage)).GetResponse();

    }

    #endregion

  }

}
```

Figura 25 – BDDPM: Código de teste gerado em C#

O código gerado pelo BDDPM é dividido em duas partes: (i) o código de teste em si, onde o programador/desenvolvedor do projeto irá trabalhar e; (2) o código de

comunicação com o BDDPM, esta parte inicia a partir do trecho “#region BddPluginComunication”.

A primeira parte será utilizada pelos programadores para inclusão dos códigos que, efetivamente, irão testar a aplicação. Conforme já citado anteriormente, uma execução de um teste pode ter 03 resultados: (i) sucesso, quando aquele recurso/instrução/funcionalidade foi implementado e está funcionando conforme a especificação feita pelo cliente; (ii) falha, quando ocorreu um erro no recurso/instrução/funcionalidade especificado pelo cliente, ou seja, o comportamento obtido não era o esperado e; (iii) inconclusivo, quando o teste para aquele recurso/instrução/funcionalidade não foi executado, ou implementado, ainda.

Toda as vezes que um teste é executado, a segunda parte do código envia estes resultados para a aplicação do BDDPM, de maneira que as informações sejam registradas/persistidas em banco de dados. As informações são enviadas via HTTP GET[101] (através de uma Chamada Remota de Procedimento, RPC, conforme sugerido no manual[102] de desenvolvimento de plugins do Mantis). Com isso, os clientes são notificados em tempo real sempre que houver algum resultado de teste disponível, ou seja, este conceito quebra o paradigma atual no qual os clientes não percebem, ou acompanham, o processo de teste do software do qual eles são donos (uma vez que estão comprando o desenvolvimento).

Esse recurso dá uma maior percepção aos clientes a respeito do andamento e do fluxo de desenvolvimento do projeto; além de abrir portas para negociação de quais funcionalidades poderiam ser incluídas, excluídas ou adicionadas (considerando um modelo flexível onde as entregas podem ser negociadas em cada iteração[103] do desenvolvimento). Também é uma excelente ferramenta que permite ao gestor ter um relatório que inclui os quantitativos de erros por comportamento esperado, ele pode responder as seguintes perguntas:

- **Quantas vezes um comportamento falhou antes de funcionar?**
- **Quais os motivos que levaram ao erro e, principalmente, como evita-los nas próximas iterações?**
- **A ordem em que as funcionalidades são desenvolvidas e testadas obedece à ordem de prioridade dos “Casos de Uso” criados pelos clientes?**
- **Qual o ritmo de testes da equipe do projeto?**

4.2.3. Geração do Código de Teste

O BDDPM é capaz de, a partir da descrição em Gherkin, criar o código de teste em C# para utilização dos desenvolvedores e testadores: os “Step Definitions”.

[101] O protocolo HTTP define oito métodos (GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS e CONNECT) que indicam a ação a ser realizada no recurso especificado. O GET solicita algum recurso por meio do protocolo HTTP. Além disso, também suporta o envio de dados através da URL através de pares (chave, valor) na estrutura do endereço. Ex: http://[endereço]?chave1=valor1&chave2=valor2....

[102] https://www.mantisbt.org/wiki/doku.php/mantisbt:plugins_overview

[103] Desenvolvimento iterativo é uma estratégia de planejamento de retrabalho em que o tempo de revisão e melhorias de partes do sistema é pré-definido.

Utilizando a funcionalidade de soma de 2 números em uma calculadora, a Figura 26 mostra os passos lógicos que são efetuados pelo plugin:

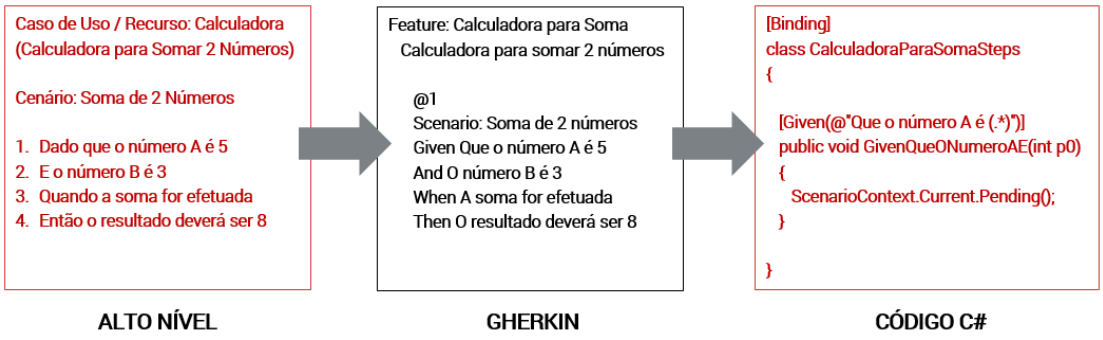

Figura 26 – Geração de Código de Teste em C# a partir dos testes do cliente

O código de geração é aberto e pode ser conferido ao fazer o download do plugin. Para que ele possa ser gerado, algumas regras devem ser obedecidas:

1. O SpecFlow, motor BDD do BDDPM, assim como outros do mercado, o do Cucumber, por exemplo, trabalha com expressões regulares[104] para identificar a qual passo o teste pertence.
2. Existem palavras reservadas que identificam o início de um passo: dado, quando, então, etc. Estas palavras reservadas são convertidas para o inglês: given, when, then, etc.
3. Estas palavras reservadas devem ser inseridas nas anotações do método exatamente com a mesma descrição / frase inserida pelo usuário. Ex: Dado que o número A é 5 → Given Que o número A é 5 → [Given(@"Que o número A é (.*)")]. Números e palavras entre aspas no Gherkin devem ser transformados em parâmetros. Neste exemplo o 5 virou "(.*)", indicando que é um parâmetro do teste. Por expressão regular, estas informações são mapeadas de maneira que se possa saber a qual passo elas pertencem.
4. Os nomes dos métodos devem ser formados a partir da concatenação da descrição daquele passo. Após a concatenação a assinatura[105] do método passa por um tratamento para deixa-lo no padrão CammelCase[106]. Assim como no passo 3, números e palavras entre aspas no são, novamente, transformados em parâmetros.
5. O corpo de cada método é sempre formado pelo código ScenarioContext.Current.Pending(), indicando que aquele teste ainda não foi implementado.

[104] Em ciência da computação, uma expressão regular provê uma forma concisa e flexível de identificar cadeias de caracteres de interesse, como caracteres particulares, palavras ou padrões de caracteres.
[105] Nome do método.
[106] CamelCase é a denominação em inglês para a prática de escrever palavras compostas ou frases, onde cada palavra é iniciada com Maiúsculas e unidas sem espaços.

4.2.4. Resultado dos Testes

O resultado dos testes pode ser acompanhado em tempo real pelo cliente, isto é, sempre que um teste for executado no Visual Studio, os resultados são enviados automaticamente para o servidor do BDDPM no momento da execução.

O mesmo resultado (Figura 27) obtido, no Visual Studio, pelo testador do projeto será enviado para o cliente (Figura 28) que terá acesso ao motivo das falhas, caso o teste falhe. Esta informação é útil, tanto para acompanhamento, quanto para que o cliente saiba quais funcionalidades foram implementadas e testadas com sucessos, quanto as que estão com falhas e as que ainda não foram incluídas em seu sistema.

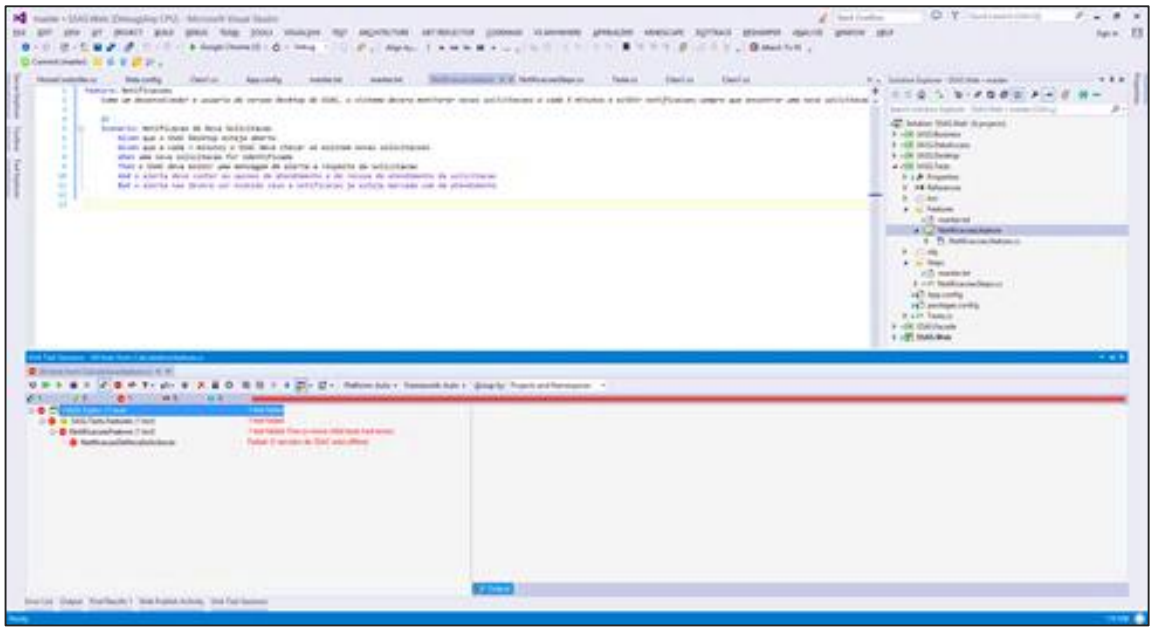

Figura 27 – Resultado de um teste no Visual Studio

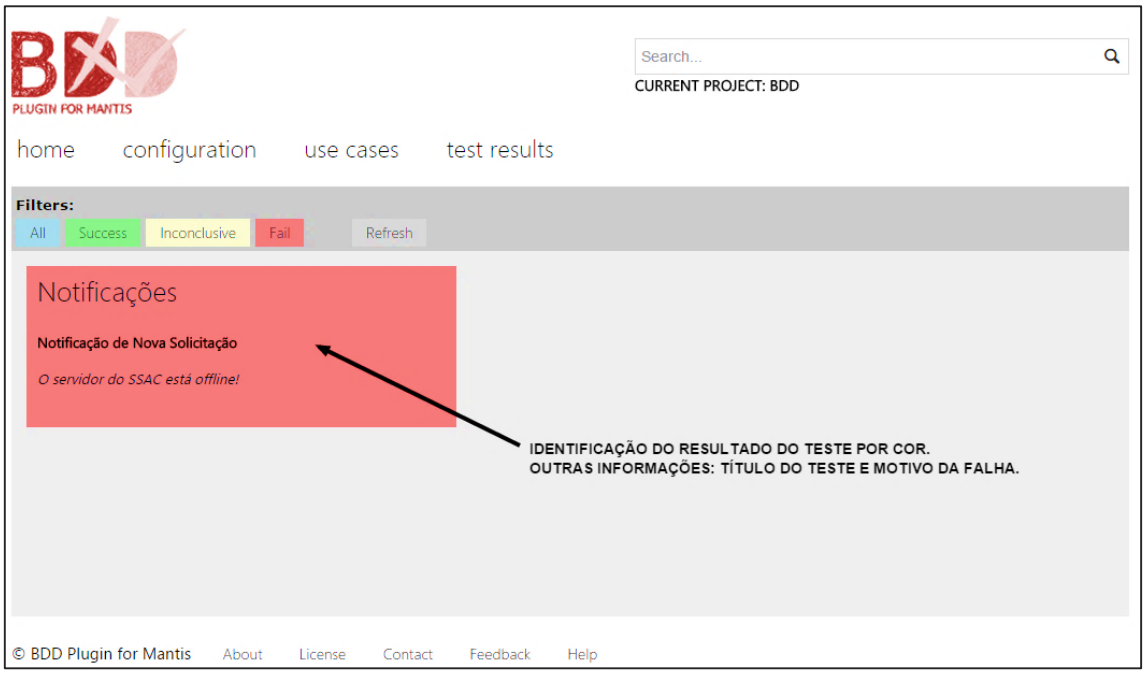

Figura 28 – Resultado de um teste no BDDPM

Capítulo 5 – Aplicação do GQM e Análise dos Resultados

O BDDPM teve todo seu ciclo de vida avaliado desde o primeiro capítulo deste documento. Do Capítulo 1 até o Capítulo 4 a ferramenta foi justificada, seu contexto de utilização foi apresentado, seus objetivos foram delineados, o cenário do BDD no mercado foi avaliado, a metodologia de desenvolvimento do BDDPM foi abordada e seus recursos foram demonstrados através de um caso de uso real na Paraíba Previdência, um órgão do governo estadual da Paraíba.

Conforme citado anteriormente, a ferramenta seria avaliada através da técnica GQM (Goal-Question-Metrics) que é alvo de estudo deste Capítulo, bem como os resultados oriundos da aplicação da mesma. A Seção 5.1 faz uma breve análise sobre o contexto para avaliação do BDDPM. A 5.2 discorre sobre o GQM. Na sequência, na Seção 5.3, é mostrado como o GQM foi utilizado para avaliação do BDDPM; e na 5.4 é mostrado o desfecho da aplicação do GQM com a avaliação dos resultados obtidos. A Seção 5.5 faz uma avaliação geral da aplicação da técnica no projeto.

5.1. Um Breve Contexto

A busca pela melhora do processo de desenvolvimento de software é algo recente. NÃO SE TRATA DO PROCESSO de desenvolvimento de software que foi iniciado entre 1842 e 1843 quando Ada Augusta Byron King, a Condessa de Lovelace, atualmente conhecida como Ada Lovelace, escreveu o primeiro algoritmo[107] para ser processado por uma máquina, a máquina analítica de Charles Babbage[108]. O objetivo deste primeiro programa de computador era calcular os números de Bernoulli[109] ("The Universal History of Computing" [Georges Ifrah, 2001]). SE TRATA DA BUSCA pela melhoria da “arte” de desenvolver softwares, o que, consequentemente, **envolve a avaliação e medição do mesmo**. Medição é um processo pelo qual números e símbolos são ligados a atributos de entidades do mundo real de tal maneira que os

[107] Um algoritmo é uma sequência finita de instruções bem definidas e não ambíguas, cada uma das quais pode ser executada mecanicamente em um período de tempo finito e com uma quantidade de esforço finita. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa. Sua implementação pode ser feita por um computador, por outro tipo de autômato ou mesmo por um ser humano. Diferentes algoritmos podem realizar a mesma tarefa usando um conjunto diferenciado de instruções em mais ou menos tempo, espaço ou esforço do que outros.

[108] Charles Babbage foi um cientista, matemático, filósofo, engenheiro mecânico e inventor inglês que originou o conceito de um computador programável. Ele é mais conhecido e, de certa forma, referenciado como o inventor que projetou o primeiro computador de uso geral, utilizando apenas partes mecânicas, a máquina analítica. Ele é considerado o pioneiro e pai da computação. Seu invento, porém, exigia técnicas bastante avançadas e caras na época. Apenas em 1991, o museu de Ciência de Londres criou a máquina diferencial de Babbage provando que suas teorias estavam corretas e teriam funcionado.

[109] Na matemática, os números de Bernoulli são sequências de números racionais com profundas conexões na teoria dos números. São definidos como os coeficientes da Expansão de Taylor:

$$\sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = \frac{x}{e^x - 1}$$

descrevem de acordo com um conjunto claro de regras ("Software Metrics, a rigorous and practical approach" [Fenton, Pfleeger, 1996]). O resultado numérico é chamado de medição e pode ser aplicado tanto ao desenvolvimento de software como ao produto de software.

De acordo com o livro "Engenharia de Software" [Ian Sommerville, 2011], esta busca começou em na década de 60, quando o termo Engenharia de Software tornou-se conhecido após uma conferência em 1968, que abordou as dificuldades e "armadilhas" de projetar sistemas complexos, algo que foi denominado de "Crise do Software". Antes disso não havia preocupação com adoção de boas práticas. A qualidade tinha uma aparência exagerada, uma perda na corrida pelo lucro e pelo pioneirismo; o rápido crescimento do poder computacional das máquinas tornou possível aplicar computação em tarefas cada vez mais complicadas, e a busca das empresas por QUANTIDADE reduziu os cuidados para um projeto de QUALIDADE. De lá para cá surgiram diversos métodos e técnicas para melhoria e avaliação de software e do processo de desenvolvimento de software.

O principal fator motivacional para que fábricas de software, desenvolvedores, gerentes, etc., se preocupem com estes métodos e técnicas é que a qualidade do produto final depende diretamente da qualidade do processo de software adotado. Um produto final de qualidade entrega VALOR ao cliente, isto é, utilidade e garantia (utilidade se relaciona com O QUE o cliente deseja, e garantia com o COMO o cliente deseja).

De lá para cá diversas normas e modelos surgiram com os objetivos de medir, avaliar e melhorar o processo de ciclo de vida do software: GQM[110] (1994), ISO/IEC 12207[111] (1995), ISO/IEC 15504[112] (1997), CMMI[113] (2002), etc. No paper[114] "The Goal Question Metric Approach" [Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach; 1994] é dito que:

"*Como em toda disciplina de engenharia, o desenvolvimento de software requer mecanismos de* ***medições para avaliação e feedback****. Nesse sentido,* ***a medição torna-se um mecanismo de auxílio*** *corporativo criado para* ***ajudar a responder a uma série de questões*** *associadas à consolidação de qualquer processo de software. Ajuda a suportar o planejamento do projeto, determinar as* ***forças e fraquezas dos processos e produtos*** *da empresa, provê um meio racional para adotar e refinar técnicas, permite* ***avaliar a qualidade dos processos e produtos****, etc. A medição também ajuda, no decorrer do projeto, acompanhar seu progresso e adotar medidas corretivas quando necessário, inclusive com* ***avaliação de impacto*** *da ação, ou ações, tomadas.*"

[110] http://www.cs.umd.edu/~mvz/handouts/gqm.pdf
[111] http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43447
[112] http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=37458
[113] http://cmmiinstitute.com
[114] Para a ABNT (1989) paper é um pequeno artigo científico, elaborado sobre determinado tema ou resultados de um projeto de pesquisa para comunicações em congressos e reuniões científicas, sujeitos a sua aceitação por julgamento.

Ainda de acordo com Basili e seus colegas, uma medição, para ser efetiva, deve ser **passível de aplicação em todo o ciclo de vida[115]** de produtos, processos e recursos. A proposta do GQM engloba esta perspectiva. A técnica vem sendo utilizada a décadas (desde a década de 70). Inicialmente foi utilizada pela NASA, mais especificamente pelo Centro de Voos Espaciais de Goddard[116], quando Dr. David Weiss e Victor Basili de depararam com um problema no qual tentavam entender os tipos de mudanças que eram realizados nos projetos de software da instituição. Mudanças eram comuns, mas eles se fizeram a seguinte pergunta: “Como é possível decidir o que deve ser medido para se atingir os objetivos?”. Para trabalhar no problema, eles decidiram levantar todos os objetivos (Goal) juntamente com uma série de questões (Question) relacionadas aos objetivos. Os questionamentos tornavam os objetivos mais específicos e claros, e era mais fácil levantar métricas (Metric) relevantes para medi-los. Surgia então, o GQM. "The Goal/Question/Metric Method: a practical guide for quality improvement of software development" [Rini van Solingen, Egon Berghout; 1999]

Com o passar dos anos, o GQM evoluiu e passou a ser utilizado em outros ambientes e contextos, inclusive com contribuições significativas do Victor Basili que publicou diversos artigos sobre estratégias de utilização da técnica. O GQM define um modelo de medição baseado em métricas para avaliação. Este modelo parte de metas e objetivos organizacionais (e/ou de projeto), estabelecidos e bem claros. Uma grande vantagem da técnica é que, embora ela tenha sido concebida para avaliação/medição de software, seus conceitos também podem ser aplicados em uma situação onde deseja-se avaliar se objetivos foram alcançados.

O BDDPM enquadra-se em um perfil que pode ser avaliado com o GQM: encontra-se na fase de introdução de seu ciclo de vida e foi criado com objetivos claros e específicos. A técnica foi utilizada para "medir" o BDDPM. As próximas seções descrevem o funcionamento da metodologia, bem como os resultados obtidos.

5.2. A Técnica do Goal-Question-Metric (GQM)

Conforme visto na anteriormente, O GQM é um modelo para definir, implantar, medir, analisar e melhorar processos/produtos/recursos. O foco primário da metodologia relaciona-se ao ambiente de desenvolvimento de software, podendo ser, entretanto, utilizado em outros contextos. De acordo com Victor, o GQM parte do pressuposto de que, para que uma empresa consiga medir objetivamente, é necessário que ela primeiro estabeleça metas e objetivos para si mesma e para seus projetos, para, em seguida, levantar um conjunto de dados que definam estes objetivos de maneira operacional para, por fim, estabelecer um framework para interpretar os dados de acordo com as metas/objetivos.

[115] O modelo de ciclo de vida de um recurso/produto/serviço é uma representação de sua história (completa) através de suas diversas fases: concepção, introdução, crescimento, maturidade e declínio.
[116] http://www.nasa.gov/centers/goddard/home

Em linhas gerais, o GQM permite entender um conjunto de informações necessárias a organização/projeto/processo/produto/recurso. Estas informações devem, sempre que possível, ser quantificadas de maneira a permitir análises sob o ponto de vista do cumprimento, ou não, dos objetivos. A estratégia parte dos objetivos, passando pelo levantamento de questões para, enfim, estabelecer métricas (Objetivos → Questões → Métricas / Goals → Questions → Metrics). Daí vem seu nome: o GQM, que consiste de quatro fases (ilustração da Figura 29):

- **Fase de Planejamento**: fase na qual é escolhido um projeto para medição. Ele deve ser definido, caracterizado e planejado. O resultado desta fase é o plano do projeto.
- **Fase de Definição**: nesta fase o programa de medição é definido (objetivos, questionamentos e métricas). Algumas hipóteses também podem ser formuladas. Tudo deve estar documentado.
- **Fase de Coleta de Dados**: os dados do projeto devem ser colhidos nesta fase.
- **Fase de Interpretação**: na última fase os dados coletados na fase anterior são processados de acordo com as métricas estabelecidas e transformados em resultados de medição que devem responder às perguntas estabelecidas na segunda fase. Depois verifica-se se as respostas atendem aos objetivos.

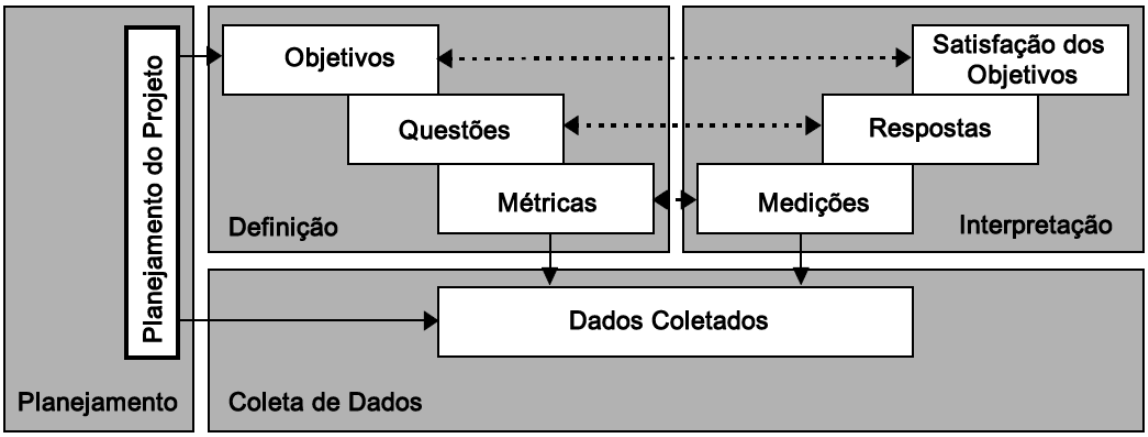

Figura 29 – Fases do GQM [Rini van Solingen et al, 1999]

Na fase de planejamento são definidos todos os detalhes e requisitos necessários para permitir que a aplicação do GQM tenha sucesso. Esta etapa inclui estudos, treinamentos, planejamento do projeto, etc. A segunda fase envolve as etapas que dão nome a metodologia do GQM, isto é, a identificação do(s) objetivo(s), a criação das perguntas emanadas dos objetivos, levantamento de métricas relacionadas e expectativas (hipóteses) sobre os questionamentos. Ao final desta fase (segunda), as medições já podem ser iniciadas, entretanto, elas precisam de uma fonte de informação, que vem da fase de coleta de dados. A fase da coleta de dados é de extrema importância, pois, quanto melhor a informação colhida, quanto mais significativa ela for, melhores serão os resultados da técnica. Além disso, a terceira fase é a maior das fases, em termos de duração: ela inicia após o final do planejamento e dura o projeto todo. Informações são colhidas o tempo todo, novas

informações são geradas e analisadas novamente. Existem diversas formas de coletar dados, o importante é que tudo deve ficar registrado. Na última etapa, por fim, os resultados das medições obtidos através das métricas são utilizados para responder as perguntas e avaliar se os objetivos foram atendidos.

O paradigma do GQM pode ser dividido em três níveis e está representado na Figura 30:

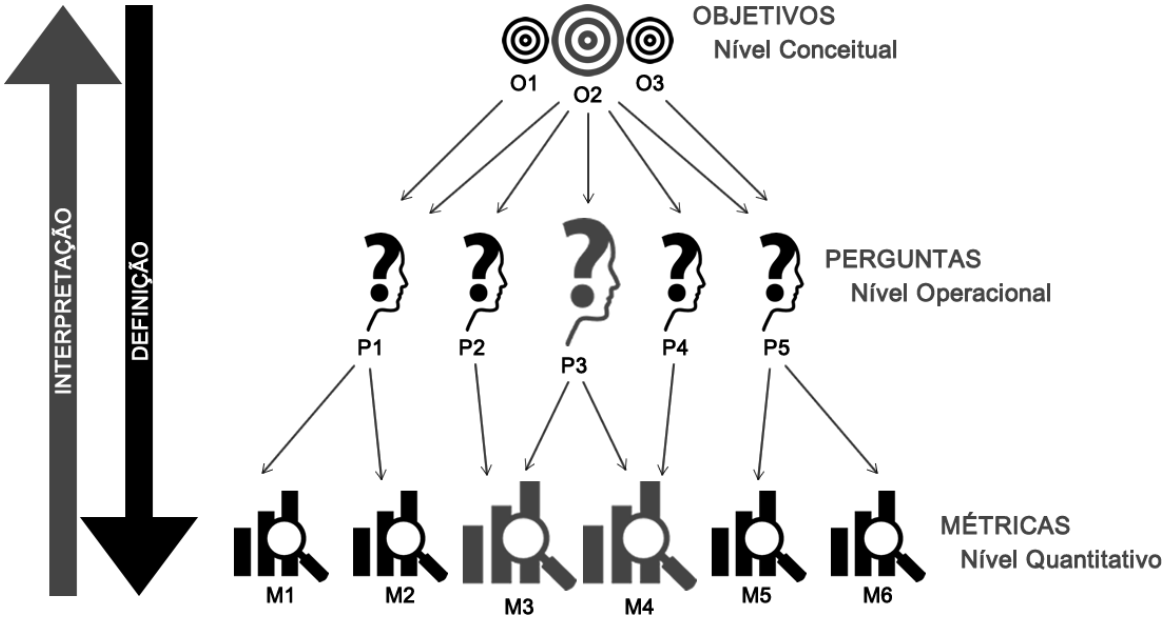

Figura 30 – Paradigma do GQM

- **Nível Conceitual (Objetivos)**: Objetivos são definidos para um objeto (produto/processo/recurso) por uma infinidade de razões, levando em consideração vários modelos de qualidade, sob diferentes pontos de vista e relativos a um cenário particular. Um objetivo especifica um motivo de medição, o objeto de medição, uma questão a ser medida e o ponto de vista de análise da medida. Este nível é chamado de conceitual por é aqui onde são desenvolvidos os conceitos a respeito de onde se quer chegar. Não se pode medir o que não é conhecido.
- **Nível Operacional (Questões/Perguntas)**: Um conjunto de questões é utilizado para caracterizar o modo pelo qual um objetivo específico pode ser atingido, além de caracterizar o objeto medido (produto/processo/recurso) de maneira qualitativa a partir de uma determinada perspectiva. Cada questão deve definir seu objetivo de maneira operacional, ou seja, envolvendo uma operação concreta para facilitar a interpretação, dado que os objetivos são definidos de forma mais conceitual e abstrata. Por isto este nível é chamado de nível operacional.
- **Nível Quantitativo (Métricas)**: As métricas identificam as medidas necessárias para responder as questões. Essas medidas são associadas à cada questão de modo a respondê-las de forma quantitativa. As métricas podem ser objetivas, quando dependem apenas do objeto medido independentemente de perspectivas de análise; e podem ser subjetivas, quando dependem, tanto do objeto medido, como da

perspectiva sob o qual o objeto está sendo analisado. Um exemplo de uma métrica objetiva é a quantidade de páginas de um documento: independentemente do ponto de vista de análise, o número é fixo. Por outro lado, existem métricas que podem variar dependendo do ponto de vista da análise; a métrica "qualidade do documento" obtida através de notas (variando de 1 a 10, por exemplo) atribuídas por avaliadores, é uma métrica subjetiva, pois cada avaliador pode atribuir uma nota diferente, pois podem analisar um documento de maneiras distintas. Independentemente do tipo da métrica, todas elas quantificam um objeto, por isto este nível é chamado de nível quantitativo.

O GQM utiliza uma abordagem TOP-DOWN (de cima para baixo) para definição dos objetivos, questões e métricas; nesta ordem. A análise e interpretação dos resultados segue o processo inverso, ou seja, através de uma abordagem BOTTOM-UP (de baixo para cima) com utilização das métricas para medição e resposta às questões e, consequentemente, avaliação de atendimento dos resultados. É importante ressaltar que a relação entre objetivos e questões; e entre as questões e as métricas, não é de UM para UM. Um objetivo pode levar a um ou mais questionamentos. Cada questionamento pode se relacionar a um ou mais objetivos e ter uma ou mais métricas de avaliação. Cada métrica pode medir uma ou mais questões. Não podem existir objetivos, questões, ou métricas soltas; isto é:

- Um objetivo deve ter, ao menos, uma questão associada.
- Uma questão deve estar associada a, pelo menos, um objetivo e uma métrica.
- Uma métrica deve estar associada a, pelo menos, um objetivo.

De maneira extremamente simplificada, um exemplo da utilização do GQM poderia ser resumido da seguinte maneira:

1. Uma fábrica de software está com o seguinte problema: atualmente, cada software que ela entrega para o cliente tem uma taxa de defeitos em torno de 5%, ou seja, 95% das funcionalidades do sistema estão corretas, porém 5% delas são entregues com falhas. O cliente desta fábrica propôs um novo acordo de nível de serviço (SLA[117]) onde essa taxa de erro deverá ser reduzida para 2%.
2. Os gestores da organização se reuniram e propuseram duas novas metodologias de desenvolvimento e teste de software que poderiam resolver o problema: a metodologia Alfa e a metodologia Beta.
3. Para avaliar as propostas e optar pela melhor delas, sugeriu-se a utilização do GQM. Os Stakeholders se reuniram para fazer o planejamento do projeto (Fase de Planejamento) para aplicação do GQM.

[117] É uma documentação sobre Service Level Agreement (SLA) ou Acordo de Nível de Serviço. Trata-se de um documento detalhado que define os padrões de um nível de serviço e a relação entre duas partes: solicitador e o executor. É um instrumento para a gestão das expectativas do cliente. Sua meta é definir uma estrutura para a gestão da qualidade e quantidade dos serviços entregues e, por consequência, atender à demanda dos clientes a partir de um entendimento claro do conjunto de compromissos.

4. Na Fase de Definição, que envolve os três níveis do GQM, chegaram às seguintes definições utilizando uma abordagem TOP-DOWN:
    a. Objetivo: Reduzir a taxa de defeito do software de 5% para 2%.
    b. Pergunta: Qual a taxa de defeito do software, quando desenvolvido utilizando a metodologia?
    c. Métrica: Taxa de defeito do software
5. Na fase de coleta de dados as duas metodologias foram implantadas e os softwares foram avaliados quanto a taxa de erros. Os softwares desenvolvidos com a metodologia Alfa tiveram uma taxa de erro de 6% e os que foram desenvolvidos com a metodologia Beta tiveram uma taxa de erro de 1,5%.
6. Na fase de avaliação, as métricas medidas (6% e 1,5%) tornaram-se respostas para cada uma das metodologias. Ao checar o cumprimento do objetivo, constatou-se que a metodologia Alfa falhou (6% responde à pergunta, mas não atinge o objetivo); e a metodologia Beta foi aceita, pois 1,5% responde à pergunta e atinge o objetivo.

5.3. O BDDPM segundo o GQM

Antes de iniciar a descrição da metodologia de avaliação e medição do BDDPM com o GQM, é extremamente importante ressaltar que ela não foi feita para avaliar o processo de desenvolvimento do plugin, mas para avaliar (com objetivos acadêmicos) uma versão finalizada do produto colocado em produção dentro de uma empresa. Nesse sentido, o tamanho da equipe de avaliação, o tempo de avaliação e os custos envolvidos diferiram significativamente dos cenários apresentados na anteriormente. Além disso, não é objetivo deste Capítulo discutir sobre problemas e desafios encontrados, o que será feito no Capítulo 6.

Esta Seção irá, portanto, se ater apenas à explanação da forma como o GQM foi aplicado (dentro das condições do projeto) para avaliação/medição do BDDPM, sem detalhar as problemáticas encontradas.

A equipe de avaliação do BDDPM foi composta por três integrantes: a equipe desenvolvedora do SSAC. O tempo da avaliação foi de dois meses. Excetuando-se os esforços de trabalho, a aplicação da metodologia não implicou em custos para a PBprev e/ou os envolvidos. Todo o processo foi aplicado com objetivos acadêmicos dentro de um projeto real que já era previsto pela Paraíba Previdência dentro da normalidade de utilização de recursos humanos e materiais, portanto não houve preocupação em registrar e/ou acompanhar quaisquer aspectos financeiros e/ou humanos e/ou de infraestrutura emanados/oriundos da aplicação do GQM.

A literatura propõe várias formas de trabalho para aplicação do GQM, todas com os mesmos objetivos. Rini van Solingen e Egon Berghout sugerem uma abordagem em forma de checklist[118] no livro "The Goal/Question/Metric Method: a practical guide

[118] Checklist é uma palavra em inglês que significa "lista de verificações". Esta palavra é a junção de check (verificar) e list (lista). Uma checklist é um instrumento de controle, composto por um conjunto de condutas, nomes, itens ou tarefas que devem ser lembradas e/ou seguidas. Uma checklist pode ser aplicada em várias atividades

for quality improvement of software development. Christiane Gresse, em "Utilização do GQM no Desenvolvimento de Software" sugere um modelo de fases e etapas. A abordagem utilizada no BDDPM mescla as duas sugestões na consecução dos trabalhos e atividades, ou seja, a equipe seguiu as fases padrão do GQM e mantinha um checklist interno que devia ser satisfeito antes de se passar para a próxima etapa.

5.3.1. Fase de Planejamento

O projeto tem início na primeira fase do GQM, a fase de planejamento, cujos principais objetivos são coletar toda informação necessária para um sucesso da introdução do GQM, estratégias para análise e interpretação dos resultados, bem como para preparar e motivar os membros da equipe no programa de mensuração. O final da fase de planejamento tem como resultado a confecção do plano de projeto que descreve os procedimentos, cronogramas e objetivos da medição, define (precisamente) as medidas, suas justificativas e como serão utilizadas, além de prover uma base para promoção e aceitação pela gestão O plano de projeto contém, ainda, um plano de treinamento para os envolvidos.

De acordo com a literatura, a fase de planejamento pode ser dividida nas seguintes etapas:

1. Estabelecimento da equipe GQM
2. Seleção do objeto de avaliação
3. Estabelecimento da equipe de avaliação do projeto
4. Criação do plano de projeto
5. Treinamento e promoção

5.3.1.1. Estabelecimento da Equipe GQM

As boas práticas de aplicação do GQM pregam que, sem uma equipe independente responsável pela continuidade do programa de medição, as atividades envolvidas no processo tendem a falhar. Quando deadlines surgem, as pessoas tendem a dar menos atenção as atividades de medição, a não ser que um time GQM independente seja estabelecido [Basili et al, 1994].

Neste sentido, no projeto de avaliação do BDDPM, a equipe GQM foi constituída com um único integrante: Alvaro Magnum, Analista de Sistemas e criador do BDDPM. Estatísticas comprovam que, quanto maior o nível de confiança mútua e cooperação entre a Equipe GQM e a Equipe de Avaliação, maiores as chances de sucesso no resultado ("Assessing feedback of measurement data: Schlumberger practices with reflection to theory" [Solingen et al, 1997]). Este fato (do relacionamento) se concretizou no projeto de avaliação do BDDPM, uma vez que todos os envolvidos se conheciam profissionalmente, pessoalmente e com anos de experiência de trabalho em equipe.

5.3.1.2. Seleção do Objeto de Avaliação

O objetivo deste trabalho sempre foi muito claro: propor um conjunto de recursos para facilitar a adoção do BDD no mercado; recursos estes que foram inseridos em

uma ferramenta que foi avaliada dentro de um ambiente de produção real através da técnica GQM. A ferramenta criada recebeu o nome de BDDPM (Behavior Driven Development Plugin for Mantis).

5.3.1.3. Estabelecimento da Equipe de Avaliação do Projeto

A equipe de avaliação do BDDPM foi composta por três integrantes: a mesma equipe desenvolvedora do SSAC:

- Daniel de Oliveira – Coordenador, Cliente e Analista do SSAC
- 02 Estagiários Desenvolvedores/Testadores

As atividades de medição do projeto foram realizadas por esta equipe. O sucesso de uma avaliação depende fortemente de sua metodologia, entusiasmo e motivação; sendo de fundamental importância a atuação do time GQM para treinamento, motivação e monitoração contínua da equipe (descritos na Seção 5.3.1.5).

5.3.1.4. Criação do Plano de Projeto

A criação do Plano de Avaliação do Projeto foi feita pela Equipe GQM, ou seja, Alvaro Magnum. Foi estabelecido um cronograma de dois meses para avaliação do BDDPM quanto aos seus objetivos. Embora a aplicação do estudo já tivesse sido autorizada pela PBprev para fins acadêmicos; para estimular o órgão, bem como a equipe de avaliação do projeto, que também é uma equipe de desenvolvimento dentro da Paraíba Previdência, foi feita uma apresentação geral do BDDPM como uma ferramenta que poderia melhorar o processo de testes dentro da instituição através da utilização do Behavior Driven Development conforme proposto pelo plugin. Caso os resultados da avaliação fossem promissores, toda a equipe já estaria treinada e apta a utilizar o BDDPM dentro dos projetos do órgão e os próprios clientes dos produtos de software a serem criados poderiam participar do processo de desenvolvimento com a criação dos testes de aceitação para os mesmos.

Como o próprio gestor de TI da PBprev (Alvaro Magnum) participou de todo o projeto, tanto como proponente, como acompanhador da avaliação, como desenvolvedor, como gestor e como analista; os avanços e resultados eram registrados e propagados semanalmente junto a alta gestão do órgão. Também foi definido que o BDDPM seria utilizado em um projeto real dentro da instituição e seria avaliado nessas condições. Por se tratar de uma primeira aplicação da metodologia do GQM, pelo curto prazo para aplicação da avaliação e pelas incertezas dos riscos envolvidos em relação ao GQM, ficou definido que todo e qualquer problema seria tratado sob demanda, ou seja, a medida que acontecesse; seguindo, mais ou menos, o modelo proposto pela técnica do Just in Time (JIT[119]), isto é, para reduzir os custos de tempo relacionados ao levantamento de possíveis problemas que pudessem ocorrer no projeto, estabeleceu-se que as decisões seriam tomadas na hora em que

[119] Just in Time é um sistema de administração da produção que determina que nada deve ser produzido, transportado ou comprado antes da hora certa. O termo “just in time”, vem do inglês e significa na hora certa. O sistema "just in time" pode ser aplicado em qualquer organização, e auxilia na redução de custos.

ele ocorressem. Esta seria a forma de gestão dos riscos. Por fim, ficou estabelecido um cronograma de dois dias para o treinamento da equipe de avaliação do projeto em cima do GQM e que todas as comunicações se dariam de forma oral, através de reuniões e encontros, para evitar ruídos[120] de comunicação.

5.3.1.5. Treinamento e Promoção

Antes de iniciar as atividades de avaliação do BDDPM, houve um treinamento de dois dias em relação às práticas do GQM bem como a apresentação da proposta do BDDPM e os motivos de sua avaliação. Foi elaborada uma estratégia especificamente voltada para a avaliação do projeto do BDDPM, com coleta de dados diárias e anotações de opiniões pessoais (inclusive com estímulo a opiniões colaborativas para melhoria da ferramenta com possibilidade de inclusão da funcionalidade na mesma) a respeito da utilização do sistema. Os princípios de medição, o paradigma GQM e a metodologia GQM foram explanados.

Os papéis de cada membro foram definidos a importância de cada integrante foi evidenciada a partir das atividades delegadas. Com uma equipe pequena a definição de papéis foi simplista: todos fariam tudo e participariam de todos os processos e fases do GQM. O único ponto que estava fixo e definido era que: (i) o BDDPM estava criado; (ii) seu objetivo era facilitar a adoção do BDD dentro das organizações, permitindo criação dos testes pelos próprios clientes de produtos de software e (iii) o BDDPM precisava ser avaliado quanto aos seus objetivos. Todos os processos restantes, como estabelecimento das metas a partir dos objetivos, levantamento de métricas, coleta de dados e avaliação dos resultados seriam feitas por todos. Este processo se mostrou bem produtivo e eficiente, com reuniões regulares, brainstorming[121] eficaz e alinhamento da equipe com os objetivos e atividades.

Em relação a promoção dos resultados, ela acontecia em tempo real, uma vez que o principal interessado neles participou ativamente de todas as etapas de todos os projetos: criação do BDDPM, criação do SSAC com utilização do BDDPM, avaliação do BDDPM, criação de dissertação sobre o BDDPM e sua avaliação.

5.3.2. Fase de Definição

A segunda fase de aplicação do GQM é a fase de definição. É nesta fase onde o método de avaliação é formalmente definido. As metas a serem alcançadas como GQM são definidas com base nos objetivos da organização/empresa e do projeto. Esta fase termina com 03 entregas: (i) o plano GQM, (ii) o plano de medição e; (iii) o plano de análise, todos contendo informações pertinentes ao programa de medição.

A fase de medição pode ser dividida, de acordo com a literatura, em 11 etapas:

[120] Ruído são todas as interferências que prejudicam o entendimento da mensagem pelo receptor durante o processo da comunicação.

[121] Brainstorming, ou "Tempestade de Ideias", é o nome dado à uma técnica grupal – ou individual – na qual potencialidades criativas são exploradas e colocadas a serviço de objetivos pré-determinados. Em termos simples, trata-se de uma reunião de trabalho em que as ideias devem fluir livremente e sem compromisso para que a inovação possa emergir.

1. Definição do(s) objetivo(s) da medição
2. Revisar ou produzir modelos de processos de software[122]
3. Condução de entrevista GQM
4. Definição de questões/perguntas e hipóteses
5. Revisão das questões/perguntas e hipóteses
6. Definição das métricas
7. Revisão das métricas
8. Produção do plano GQM
9. Produção do plano de medição
10. Produção do plano de análise
11. Revisão dos planos

Os itens 8, 9 e 10 podem ser iniciados logo após o item 7 (revisão das métricas), e podem ser realizados em paralelo. A forma como estes passos foram implementados na avaliação do BDDPM é descrita a partir da próxima Seção. Cabe ressaltar que esta foi a última fase feita antes do início da utilização do BDDPM no desenvolvimento do SSAC. Com isso, foi possível captar as expectativas dos avaliados antes deles começarem a utilizar a ferramenta e avalia-la, de fato. As fases de coleta de dados e análise dos mesmos foram conduzidas durante o desenvolvimento do SSAC e após sua conclusão.

5.3.2.1. Definição dos Objetivos da Medição

Os objetivos da medição foram definidos por toda equipe de avaliação do BDDPM e baseiam-se nos objetivos e motivações que deram início ao BDDPM. São eles:

1. **Fácil de utilizar em termos de usabilidade**
2. **Permitir a aplicação do BDD de maneira eficiente e eficaz**
3. **Facilitar a adoção do BDD**
4. **Permitir a escrita dos testes pelos clientes**

Os objetivos 3 e 4 são o “carro chefe” do BDDPM, mas podem ser fortemente impactados pelos objetivos 1 e 2. Em resumo: a ferramenta teria que ser fácil de utilizar, até mesmo porque os próprios clientes poderão criar testes com ela; e, uma vez sendo uma ferramenta BDD com o objetivo de entregar valor para seus usuários, então teria que ser eficaz (permitir a utilização e aplicação do BDD, de fato) e eficiente (ser rápida, acelerar a aplicação da metodologia, reduzir tempos de testes, etc.). Entregar valor é entregar utilidade (o que o cliente quer) e garantia (como o cliente quer). Em qualquer projeto, os quesitos escopo, prazo e custo têm grande peso. Nesse caso escopo é o BDD e, através da facilidade, eficiência e eficácia, pode haver redução de custos e prazos; o que é bastante desejado, impulsiona a aceitação da ferramenta e, consequentemente a adoção do BDD.

[122] Etapa dispensada uma vez que não há avaliação de processos de desenvolvimento e/ou de melhoria de processos e/ou de qualidade de software, mas avaliação estrita de cumprimento de objetivos.

A equipe chegou à conclusão que, se estes 04 itens pudessem ser medidos, seus resultados poderiam dar um forte indício dos potenciais do BDDPM. Embora as questões envolvam posicionamentos subjetivos, as 04 perguntas podem fornecer métricas quantificáveis. Com isso, foram definidos os objetivos/metas da avaliação (ver Tabelas 6, 7, 8 e 9):

Objeto de Estudo	BDDPM
Objetivo	Impulsionar a aceitação e utilização do BDDPM
Enfoque	Facilidade de Uso
Ponto de Vista	Usuários do BDDPM
Contexto	Projetos de desenvolvimento de software → testes

Tabela 6 – Objetivo de medição 1: Fácil de utilizar em termos de usabilidade

Objeto de Estudo	BDDPM
Objetivo	Aplicação do BDD em testes de Software
Enfoque	Eficiência e Eficácia
Ponto de Vista	Gestores, desenvolvedores e testadores de software
Contexto	Projetos de desenvolvimento de software → testes

Tabela 7 – Objetivo de medição 2: Permitir a aplicação do BDD de maneira eficiente e eficaz

Objeto de Estudo	BDDPM
Objetivo	Adoção da técnica do Behavior Driven Development
Enfoque	Facilidade de Aplicação da técnica
Ponto de Vista	Clientes, gestores, desenvolvedores e testadores
Contexto	Projetos de desenvolvimento de software → testes

Tabela 8 – Objetivo de medição 3: Facilitar a adoção do BDD

Objeto de Estudo	BDDPM
Objetivo	Criação de testes sem necessidade de conhecimento técnico
Enfoque	Escrita de testes
Ponto de Vista	Clientes de produtos de Software
Contexto	Projetos de desenvolvimento de software → testes

Tabela 9 – Objetivo de medição 4: Permitir a escrita dos testes pelos clientes

Para futuras referências ficou estabelecido que os objetivos seriam referenciados na sequência da lista acima, recebendo um prefixo “O”, de objetivo. Sendo assim, existem 04 objetivos de avaliação: do O1 até o O4.

5.3.2.2. Condução de entrevista GQM

O objetivo das entrevistas é adquirir mais informações a respeito dos objetivos de maneira a facilitar sua compreensão e facilitar a criação das perguntas e levantamentos de métricas nas etapas posteriores. Recomenda-se entrevistar a equipe do projeto [Rini van Solingen et al, 1999] e cada uma das pessoas citadas no item “Ponto de Vista” dos objetivos levantados [Christiane Gresse, 2000]. Também recomenda-se executar entrevistas individuais para evitar influências. As entrevistas são feitas utilizando-se uma ferramenta chamada “Abstraction Sheet” que, para fins de simplicidade e objetividade; trata-se de um documento de uma única página com 04 quadrantes contendo as seguintes informações cada um (ver Figura 31):

- **Possíveis Métricas**: De acordo com a experiência do entrevistado, quais as possíveis métricas que poderiam ser utilizadas para medir os objetivos? Ex: número total de defeitos.
- **Expectativas de medição (hipóteses)**: De acordo com a experiência do entrevistado, qual a hipótese dele para a resultados de mensuração para cada métrica levantada por ele? Ex: número total de defeitos: 30.
- **Fatores de Impacto**: De acordo com a experiência do entrevistado, quais fatores podem impactar nas métricas levantadas por ele? Ex: forma como o código foi inspecionado a procura de defeitos.
- **Impactos dos Fatores**: De acordo com a experiência do entrevistado, quais os impactos que os fatores citados no ponto anterior terão nas métricas, ou seja, como elas serão influenciadas? Ex: a rigorosidade da inspeção do código influencia na quantidade de defeitos encontrados.

Antes da aplicação das entrevistas, foi feita uma reunião para que todos os envolvidos entendessem plenamente os objetivos levantados no passo anterior, pois eles seriam o foco da avaliação. Vale a pena ressaltar que nem todas as medições estarão presentes no “Abstraction Sheet”, apenas as mais importantes do ponto de vista de cada colaborador. Conforme explanado anteriormente, foi decidido que todos os integrantes da equipe de avaliação participariam de todas as etapas de aplicação do GQM. Desta maneira, o questionário foi respondido pelos três membros:

- **Daniel de Oliveira:** Analista de Sistemas, Coordenador e Cliente do Projeto SSAC, Integrante da Equipe de Avaliação do BDDPM.
- **Marcos Fernando:** Estagiário, Desenvolvedor do SSAC
- **João Paulo:** Estagiário, Desenvolvedor do SSAC

Figura 31 – Exemplo de um “Abstraction Sheet”

As entrevistas foram realizadas com todos os avaliadores e considerando os quatro objetivos do BDDPM. A Tabela 10 mostra uma parte dos resultados colhidos através da Abstraction Sheet para o objetivo de medição 1:

Objetivo de medição 1: Fácil de utilizar em termos de usabilidade
1. Daniel de Oliveira: ▪ **Possíveis Métricas:** i. Poucas funções / opções em tela. ii. Complexidade das funções / opções em tela (Nível de dificuldade). iii. Nota para nível de facilidade. iv. Nota em termos de facilidade de uso. v. Quantidade de cliques para se chegar onde quer. vi. Nota em termos de intuitividade de interface. ▪ **Expectativas de Medição:** i. Poucas funções / opções em tela: Até 4. ii. Complexidade das funções / opções em tela: 1 (3 → Mais Difícil). iii. Nota para nível de facilidade: de 2 a 3 (3 → Mais Fácil). iv. Nota em termos de facilidade de uso**:** 3 (Mais Fácil). v. Quantidade de cliques para se chegar onde quer: Até 3. vi. Nota em termos de intuitividade de interface: 3 **(**Mais Intuitivo). ▪ **Fatores de Impacto:** i. Complexidade da funcionalidade. ii. Subjetividade da facilidade. iii. Maturidade do usuário em relação a uso de Sistemas de Informação. iv. Subjetividade da Pergunta ▪ **Impactos dos Fatores:** i. Normalmente, quanto mais complexa a funcionalidade, mais complexa é a tela. ii. O conceito de fácil e difícil é subjetivo e depende do nível de conhecimento de cada pessoa. iii. Quanto maior a maturidade do usuário, mais confiável será o resultado da medição. iv. Alguns usuários podem necessitar de maiores interações (cliques) para alcançar seus objetivos. v. A valoração do atributo facilidade varia de pessoa para pessoa.

Tabela 10 – Avaliação do BDDPM: Resultado da entrevista GQM

5.3.2.3. Definição de questões/perguntas e hipóteses

Esta etapa inicia-se com o refinamento da etapa anterior para obtenção das perguntas GQM, que devem ser claras, objetivas e operacionais para que as métricas possam ser avaliadas/medidas a partir delas. Com a participação de todos os membros envolvidos, foram levantadas diversas possíveis perguntas e hipóteses. Ainda que as entrevistas tenham sido conduzidas individualmente, houve redundância de informação, provavelmente pela similaridade de perfis e pela alta expectativa em relação ao BDDPM.

Após dois dias de reuniões, a equipe de avaliação do BDDPM, com a participação de todos, chegou-se as seguintes conclusões/definições:

- A avaliação do BDDPM envolverá bastante subjetividade, ainda que colocada em termos quantitativos.
- Os integrantes utilizaram diferentes escalas de notas. Elas teriam que ser padronizadas. Estabeleceu-se uma escala crescente de 1 a 4. A nota 1 e nota 4 teriam suas descrições fornecidas nas perguntas e as notas 2 e 3

deveriam ser inferidas pela pessoa questionada. A nota 2 teria valor próximo ao da nota 1, e o valor 3 teria o valor próximo ao da nota 4. Não foi estabelecido um valor médio/intermediário pelo qual o avaliado pudesse optar. Essa foi uma opção estratégica para evitar a neutralidade da resposta. Todos os avaliados teriam que se posicionar de um lado, ou de outro, da escala.

- Na etapa anterior houve grande preocupação com o nível de conhecimento técnico dos usuários do BDDPM, mas chegou-se à conclusão que esta preocupação deve ser restrita aos clientes dos projetos de desenvolvimento de software; uma vez que o BDDPM será utilizado por gestores, desenvolvedores e testadores que deveriam possuir o conhecimento técnico necessário para aplicação do BDD. Em relação aos clientes, seria necessário apenas familiaridade na interação homem-máquina e entender a forma de utilização do BDDPM.
- Foram levantadas as seguintes perguntas (com suas respectivas hipóteses):
    1. **Em relação a facilidade de uso do BDDPM, que nota, de 1 a 4, você atribui à ferramenta? Considere uma escala onde 1 significa "Difícil de Usar" e 4 "Fácil de Usar". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fácil" e "Difícil", respectivamente.**
        - ✓ Espera-se que a grande parte das respostas atribuam notas 3 e 4 (alta expectativa).
    2. **Para utilizar o BDDPM de maneira satisfatória, qual o nível (de 1 a 4) de conhecimento técnico empregado por você? Considere uma escala onde 1 significa "Nenhum Conhecimento Técnico" e 4 significa "Muito Conhecimento Técnico". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Nenhum" e "Muito", respectivamente.**
        - ✓ Espera-se que a grande parte das respostas atribuam notas 1 e 2 (alta expectativa).
    3. **Considerando o objetivo do BDDPM de permitir a utilização da técnica do BDD em um projeto de desenvolvimento de software, bem como os recursos oferecidos pela ferramenta; classifique a interface da ferramenta atribuindo uma nota de 1 a 4. Considere uma escala onde 1 significa "Pouco Intuitiva" e 4 significa "Muito Intuitiva". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.**
        - ✓ Espera-se que a grande parte das respostas atribuam notas 3 e 4 (alta expectativa).
    4. **Considerando os recursos do BDDPM, como você classificaria sua interface considerando-a como um meio de acesso eficaz e atraente a estes recursos? Considere uma escala onde 1 significa "Baixa Qualidade" e 4 significa "Alta Qualidade". As**

notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Baixo" e "Alto", respectivamente.

- ✓ Espera-se que a grande parte das respostas atribuam notas 3 e 4 (alta expectativa).

5. **Qual a média de tempo (em minutos) para execução da fase de testes dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)**
   - ✓ Pergunta para dar suporte à avaliação da pergunta 6. Não causa impacto(s) sobre o(s) objetivo(s). Na PBprev, uma vez o sistema tendo sido desenvolvido, eram dedicados 05 dias para realização de testes. Sendo que dois dias para a equipe de testes e três dias para os usuários do sistema. Cada dia é computado com 08 horas de trabalho. Dessa maneira, tem-se: 5 x 8 x 60 = 2400 minutos.
6. **Qual a média de tempo (em minutos) para execução da fase de testes dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?**
   - ✓ Espera-se que a utilização do BDDPM traga uma redução dos tempos de testes, tanto para a equipe de testes, quanto para os usuários do sistema desenvolvido. Espera-se que os novos prazos sejam de um dia para a equipe de testes e dois dias para os usuários do sistema. Dessa maneira, tem-se: 3 x 8 x 60 = 1440 minutos. Isso representaria uma melhora de 40%.
7. **Qual a média de tempo (em minutos) de duração dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)**
   - ✓ Pergunta para dar suporte à avaliação da pergunta 8. Não causa impacto(s) sobre o(s) objetivo(s). Na PBprev, cada projeto tem uma duração que varia entre 01 e 02 (meses). Algo em torno de 45 dias, com 08 horas de trabalho diário. Sendo assim, tem-se: 45 x 8 x 60 = 21600 minutos.
8. **Qual a média de tempo (em minutos) de duração dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?**
   - ✓ Espera-se que, para uma primeira utilização do BDDPM, o tempo de desenvolvimento permaneça o mesmo, ou um pouco mais elevado, se comparado ao histórico de tempos de projetos. Entretanto, com a familiarização em relação a utilização da ferramenta, e com expectativas de redução dos tempos de testes reduzidos em até 40%, espera-se que o tempo do projeto seja reduzido em até 15%; ou seja, seriam em torno de 38 dias de desenvolvimento. Sendo assim, tem-se 38 * 8 * 60 = 18240 minutos, aproximadamente.

9. **Levando-se em consideração a utilização do BDDPM no projeto de desenvolvimento de software, como você classifica (nota de 1 a 4) o impacto dos benefícios agregados ao projeto? Considere uma escala onde 1 significa "Baixo Impacto" e 4 significa "Alto Impacto". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Baixo" e "Alto", respectivamente.**
    - ✓ Espera-se que a grande parte das respostas atribuam notas 3 e 4 (alta expectativa).
10. **Considerando a técnica do BDD, como você classificaria (nota de 1 a 4) o nível de alinhamento e atendimento do BDDPM a ela? Considere uma escala onde 1 significa "Fraco" e 4 significa "Forte". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fraco" e "Forte", respectivamente.**
    - ✓ Espera-se que a grande parte das respostas atribuam notas 2 e 3 (expectativa média). A equipe já sabe que várias funções do BDD não são incluídas na ferramenta.
11. **Qual o quantitativo médio de falhas/bugs encontrados nos produtos dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)**
    - ✓ Pergunta para dar suporte à avaliação da pergunta 12. Não causa impacto(s) sobre o(s) objetivo(s). Na PBprev, após um sistema ser colocado em produção, encontram-se, em média, 12 bugs/falhas/defeitos no mesmo (ao longo de sua vida).
12. **Qual o quantitativo médio de falhas/bugs encontrados nos produtos dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?**
    - ✓ Espera-se que a quantidade de bugs/falhas/defeitos encontrados caia para 5, uma melhora de, aproximadamente, 58% (alta expectativa).
13. **Qual o seu nível (nota de 1 a 4) de satisfação em relação a eficácia e utilização do BDDPM em um projeto de desenvolvimento de software? Considere uma escala onde 1 significa "Pouco Satisfeito" e 4 significa "Muito Satisfeito". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.**
    - ✓ Espera-se que a grande parte das respostas atribua nota 3, uma vez que alguns recursos BDD estão ausentes da ferramenta.
14. **Ao utilizar o BDDPM no projeto de desenvolvimento de software, você precisou recorrer à outras ferramentas para aplicação do BDD? Responda com Sim, ou Não.**

- ✓ Caso o projeto seja simples, espera-se que a maior parte das respostas seja “Sim”, porém, no caso de projetos maiores e mais complexos, essa quantidade pode cair drasticamente e ser superada pela quantidade de “Nãos”.

15. **Você pretende utilizar o BDDPM em outros projetos? Responda com Sim, ou Não.**
    - ✓ Espera-se que a maior parte das respostas seja “Sim”, principalmente com a possibilidade de evolução da ferramenta e inclusão de novas facilidades e recursos.
16. **Considerando a forma de disponibilização do BDDPM, como você classificaria (nota de 1 a 4) sua acessibilidade e visibilidade? Considere uma escala onde 1 significa "Pouca acessibilidade / visibilidade" e 4 significa "Muita acessibilidade / visibilidade". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.**
    - ✓ Espera-se que a grande parte das respostas atribuam notas 1 e 2 (baixa expectativa), uma vez que não previsão de disponibilização/divulgação da ferramenta em portais com grandes visitações. Ela está apenas acessível através de um repositório GIT hospedado no Bitbucket[123] (https://alvaromagnum@bitbucket.org/alvaromagnum/bddpm.git).
17. **Quantas linguagens de programação devem ser suportadas, em sua opinião, para que BDDPM tenha grande aceitação no mercado?**
    - ✓ Espera-se que as pessoas pensem nas linguagens de programação mais utilizadas no mercado e respondam com um quantitativo em torno de 05 linguagens.
18. **Qual o seu nível de dificuldade (nota de 1 a 4) para criar um teste no BDDPM? Considere uma escala onde 1 significa "Fácil" e 4 significa "Difícil". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fácil" e "Difícil", respectivamente.**
    - ✓ Espera-se que a grande parte das respostas atribuam notas 1 e 2 (alta expectativa).
19. **Quanto tempo, em média (minutos), você leva para criar um teste no BDDPM?**
    - ✓ Espera-se que este tempo varie, dependendo da complexidade do projeto e do teste a ser criado. Também existem 02 cenários. O primeiro cenário refere-se àquele no qual o cliente está concebendo os testes à medida em que usa o BDDPM. No segundo cenário, os clientes já conceberam o teste e apenas vão utilizar a ferramenta para

[123] Bitbucket é um serviço de hospedagem de repositórios para projetos que usam o Mercurial ou Git. O órgão oferece planos comerciais e gratuitos, públicos e privados.

inserção do mesmo. Considerando-se o primeiro cenário, para testes mais simples a equipe acredita que alguns poucos minutos (5 no máximo) seja suficiente para cada teste. Para projetos mais complexos, este tempo poderia chegar até 10 minutos. No segundo cenário, estes tempos passam a ser de 2 e 5 minutos, respectivamente.

Para futuras referências ficou estabelecido que as perguntas seriam referenciadas na sequência da lista acima, recebendo um prefixo “P”, de pergunta. Sendo assim, existem 19 perguntas de avaliação: da P01 até a P19.

5.3.2.4. Definição das Métricas

É nesta etapa que as métricas são definidas para prover toda informação quantitativa necessária para responder as perguntas de maneira satisfatória. As métricas podem ser consideradas como refinamento das perguntas em modelos quantitativos e/ou medições de produto. Uma vez que as métricas forem medidas, deverá haver informação suficiente para responder às questões. No projeto de avaliação do BDDPM as métricas estabelecidas foram as seguintes:

1. Facilidade de Uso (Nota).
2. Quantidade de Cliques para se Atingir o Objetivo.
3. Nível de Conhecimento Necessário para Utilização do BDDPM (Nota).
4. Interface Intuitiva (Nota).
5. Complexidade de Uso (Nota).
6. Qualidade da Interface (Nota).
7. Índice de Melhora no Tempo de Execução dos Testes (Percentual).
8. Índice de Melhora no Tempo de Desenvolvimento do Projeto (Percentual).
9. Impacto Positivo no Projeto (Nota).
10. Adequação ao BDD (Nota).
11. Relevância das Funcionalidades BDD implementadas (Nota).
12. Nível de Atendimento às Funções do BDD (Nota).
13. Índice de Melhora da Quantidade de Falhas em Sistemas (Percentual).
14. Satisfação do Cliente do Projeto (Nota).
15. Satisfação do Gestor do Projeto (Nota).
16. Satisfação do Desenvolvedor do Projeto (Nota).
17. Satisfação do Testador do Projeto (Nota).
18. Necessidade de Utilização de Ferramenta Auxiliar Externa (Sim | Não).
19. Pretensão de Utilização da Ferramenta em Outros projetos (Sim | Não).
20. Relevância dos locais/formas de distribuição do BDDPM (Nota).
21. Quantidade de Linguagens Suportadas pelo BDDPM.
22. Relevância da Forma de Divulgação do BDDPM (Nota).
23. Esforço Necessário para Criação de um Teste (Nota).
24. Tempo Necessário para Criação de um Teste (Em Minutos).
25. Facilidade de Escrita dos Testes (Nota).

A escala adotada para atribuição das notas variou de 1 a 4. Para cada métrica atribuível por nota, existe uma pergunta referente a ela que descreve a escala. Em

relação as métricas avaliadas pelas respostas “Sim” e “Não”, a avaliação seria feito após a verificação do quantitativo de pessoas que marcaram cada uma das respostas. Para futuras referências ficou estabelecido que as métricas seriam referenciadas na sequência da lista acima, recebendo um prefixo “M”, de métrica. Sendo assim, existem 25 métricas de avaliação: da M01 até a M25.

5.3.2.5. Produção do Plano GQM

O plano GQM nada mais é do que a documentação dos objetivos, perguntas e métricas estabelecidas anteriormente. Seu objetivo é servir de guia para interpretação dos dados e servir de base para as duas últimas etapas: coleta e análise de dados. O plano GQM de avaliação do BDDPM contém 04 objetivos, 19 perguntas e 25 métricas. A arquitetura do plano GQM do projeto de avaliação do BDDPM pode ser consultado através da Figura 32:

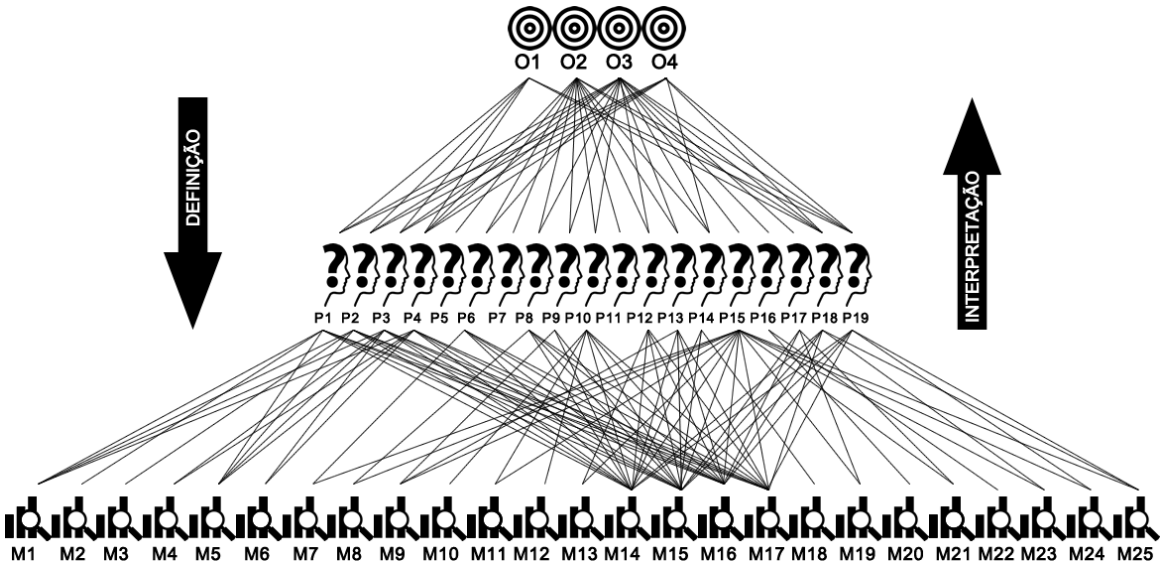

Figura 32 – Avaliação do BDDPM: Arquitetura do plano GQM

Para uma melhor visualização dos relacionamentos, da Figura 33 a 36 é possível conferir a relação “Objetivo → Perguntas → Métricas” separada por objetivo (as perguntas 5 e 7 servem apenas para dar suporte às perguntas 6 e 8, respectivamente):

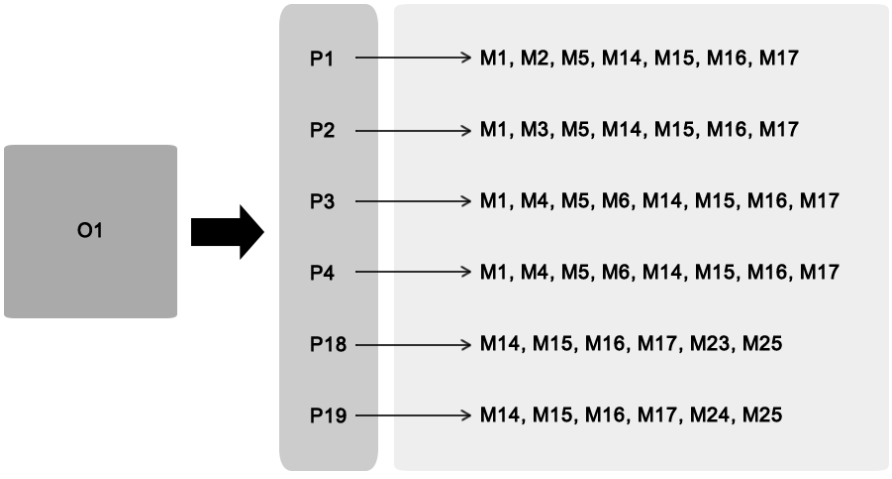

Figura 33 – Avaliação do BDDPM: Plano GQM → Objetivo 1

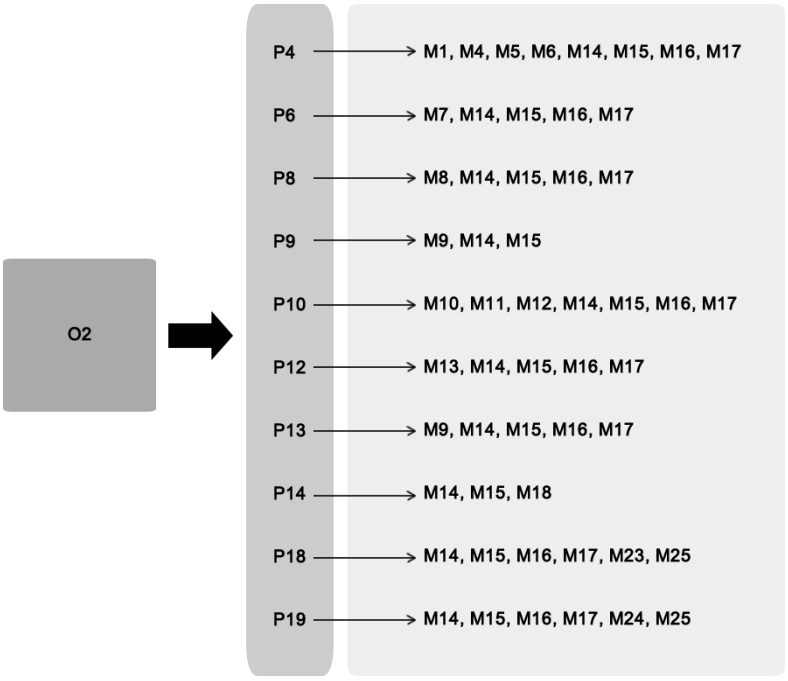

Figura 34 – Avaliação do BDDPM: Plano GQM → Objetivo 2

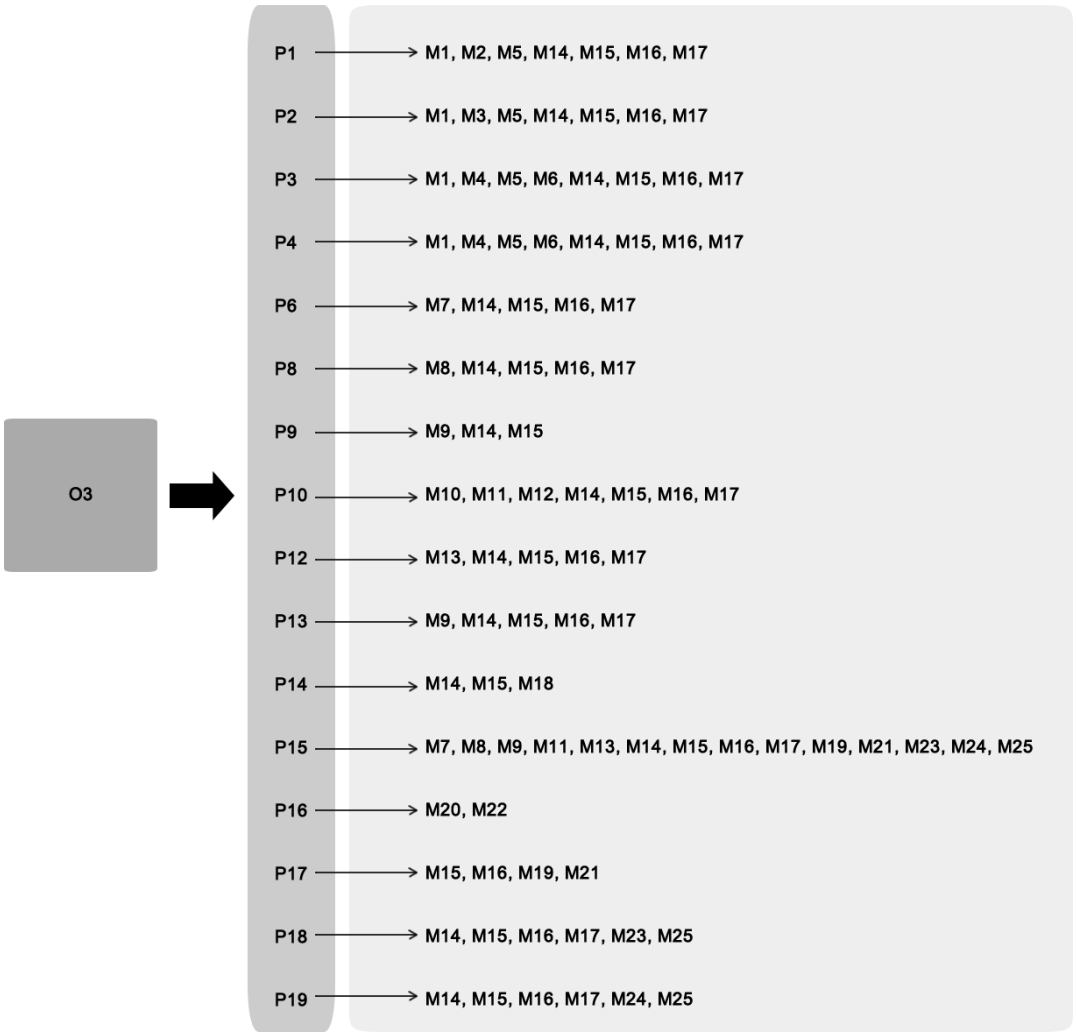

Figura 35 – Avaliação do BDDPM: Plano GQM → Objetivo 3

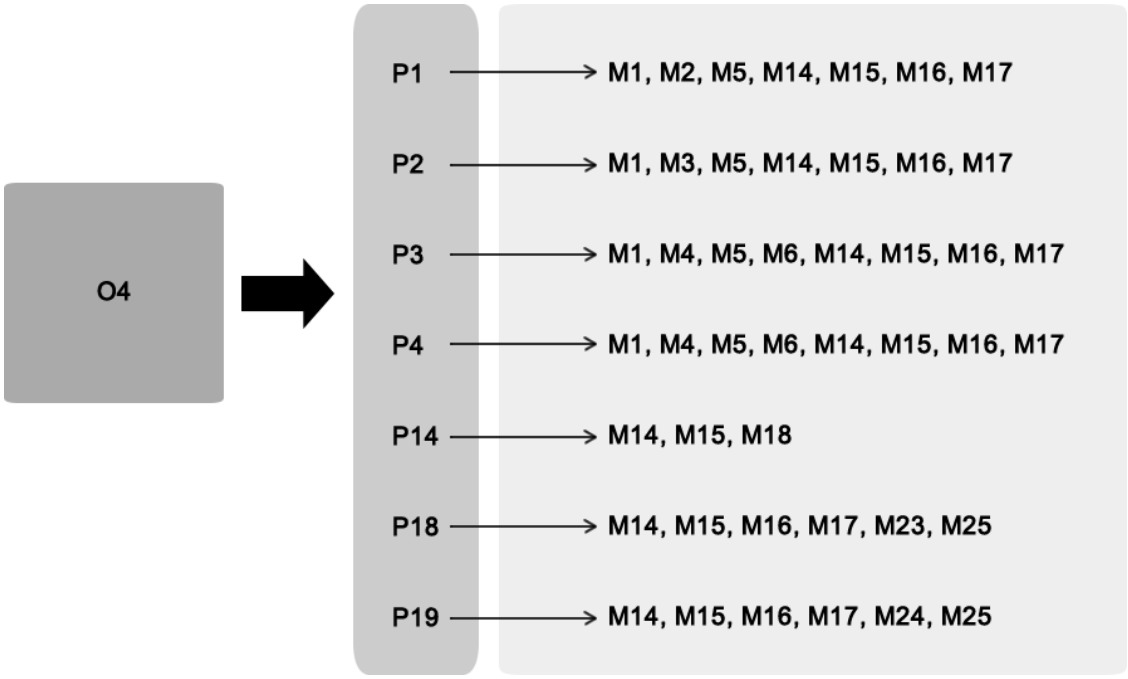

Figura 36 – Avaliação do BDDPM: Plano GQM → Objetivo 4

5.3.2.6. Produção do Plano de Medição

O principal objetivo do Plano de Medição é integrar todas as métricas e medidas no plano do projeto identificando-se o momento, a forma e a responsabilidade da coleta da informação.

No projeto do BDDPM foi estabelecido que todos os 03 integrantes da equipe de avaliação eram responsáveis por avaliar TODAS as perguntas estabelecidas. O momento da avaliação ocorreu durante o desenvolvimento do SSAC, no qual o BDDPM foi utilizado. Ao final do projeto a coleta das avaliações (medições) foram feitas através de um questionário contendo todas as perguntas. Cada integrante recebeu o questionário e respondeu às questões.

Por exemplo, supondo a primeira pergunta do questionário. A pergunta 01:

- **Está relacionada a 03 objetivos:**
  - O1 - Fácil de utilizar em termos de usabilidade
  - O2 - Permitir a aplicação do BDD de maneira eficiente e eficaz
  - O3 - Facilitar a adoção do BDD
- **Está relacionada a 07 métricas:**
  - M01 - Facilidade de Uso
  - M02 - Quantidade de Cliques para se Atingir o Objetivo
  - M05 - Complexidade de Uso
  - M14 - Satisfação do Cliente do Projeto
  - M15 - Satisfação do Gestor do Projeto
  - M16 - Satisfação do Desenvolvedor do Projeto
  - M17 - Satisfação do Testador do Projeto

O texto da pergunta um diz: “**Em relação a facilidade de uso do BDDPM, que nota, de 1 a 4, você atribui à ferramenta? Considere uma escala onde 1 significa "Difícil de Usar" e 4 "Fácil de Usar". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fácil" e "Difícil", respectivamente.**”

O avaliador deve responder à pergunta diretamente (métrica M01), mas também deve atribuir valores às outras métricas sob o ponto de vista da pergunta. Por exemplo: o foco da pergunta 1 é a “facilidade”, então o avaliador tem que atribuir um valor a cada uma das outras métrica com este enfoque até que todas as métricas relacionadas sejam medidas. Nos casos onde a métrica sugere uma perspectiva; seja do cliente, do gestor, do desenvolvedor, ou do testador; o avaliador deve-se colocar na posição de um deles e fazer a medição.

Ao final da avaliação todas as métricas estarão medidas por todos os avaliadores e sob diferentes pontos de vista. Isso fez-se necessário, pois todas as métricas relacionadas a uma pergunta têm impacto sob ela e, consequentemente, nos objetivos relacionados.

5.3.2.7. Produção do Plano de Análise

O plano de análise é um documento que simula a interpretação dos dados baseando-se nas hipóteses estabelecidas na fase de definição. Este plano servirá de guia a respeito de como os resultados devem ser analisados a fim de se alcançar um resultado. Além disso, também dá uma indicação de qual seria um possível resultado da avaliação baseando-se nas expectativas iniciais do time.

Em relação ao projeto de avaliação do BDDPM, a avaliação final se dará com a análise e consolidação dos 03 questionários respondidos, de maneira individual, pelos 03 integrantes da equipe. Como cada pergunta tem muitas métricas associadas, estabeleceu-se que a avaliação seria dada da seguinte maneira:

1. A cada métrica seria atribuído o valor “BOM” ou o valor “RUIM” de acordo com a medição (lembrando que a escala atualizada não dá margens para valores neutros).
2. Para as questões com resultados em percentual, qualquer índice superior a 10% seria classificado como “BOM”.
3. Para as questões cujas respostas fossem “SIM” ou “NÃO”, a análise seria obtida pela quantidade de votos obtidos, sendo que a maioria absoluta iria definir pelo resultado “BOM” ou “RUIM”.
4. Com todas as métricas avaliadas, obter-se-ia a quantidade de “medições boas”.
5. Caso o quantitativo de marcações obtido no passo anterior representasse, pelo menos, 70% da quantidade de métricas relacionadas à pergunta, a avaliação final seria de “ATENDE AO(S) OBJETIVO(S)”. Caso contrário, a avaliação final seria “NÃO ATENDE AO(S) OBJETIVO(S)”.
6. O mesmo processo seria aplicado para todos os objetivos com mais de uma pergunta. Caso, pelo menos, 70% das perguntas fossem respondidas satisfatoriamente, o objetivo seria considerado como

"ATENDIDO". Caso contrário, o objetivo seria considerado como "NÃO ATENDIDO".

7. Para que o objeto avaliado fosse classificado como um "SUCESSO", pelo menos 80% dos objetivos relacionados deveriam ser atendidos. Caso contrário, a classificação seria de "FRACASSO".
8. Caso fosse necessário fazer arredondamento em algum dos cálculos citados nos passos anteriores, dever-se-ia arredondar para o valor inteiro mais próximo.
9. No caso de a avaliação ser conduzida por mais de uma pessoa, a mesma regra do item 7, deveria ser aplicada, ou seja. Pelo menos 80% das pessoas deveria classificar o objeto como um SUCESSO, para que ele, de fato, esse status se concretizasse. Ou então, o resultado final seria de fracasso.

Seguindo-se estas instruções, o BDDPM seria considerado um sucesso se cumprisse com 03, dos seus 04 objetivos. Uma simulação de avaliação foi feita antecipadamente (Plano de Análise). Nela o projeto do BDDPM atingiu o índice mínimo aceitável para ser **considerado um sucesso (80)**.

5.3.3. Fase de Coleta de Dados

Esta fase se deu início com o desenvolvimento do SSAC. Durante todo o desenvolvimento do sistema da PBprev foi solicitado que os envolvidos estivessem sempre atentos a itens como: complexidade de uso, quantidade de cliques necessários para concretizar ações, etc. Cada observação e/ou detalhe importante deveria ser anotado e registrado.

O SSAC é um sistema muito simples, seu desenvolvimento não deveria ter levado mais de 30 (trinta) dias, porém, como projeto piloto do BDDPM, bem como o esforço extra para coleta e documentação dos dados para aplicação do GQM, o tempo de desenvolvimento subiu para 45 (quarenta e cinco) dias; mas acabou dentro da média de tempo dos projetos dentro da instituição. Este foi o tempo de duração da fase de coleta, que foi acompanhada todos os dias pela equipe GQM do projeto.

O fato da equipe GQM ser composta pelo chefe de todos os envolvidos na avaliação auxiliou na manutenção da motivação, além disso a integração da equipe e o coleguismo de longa data também ajudaram nesse sentido e na fluidez dos trabalhos.

No final do período, havia informações suficientes, além das próprias opiniões e experiências de cada um na utilização da ferramenta.

5.3.4. Fase de Análise de Dados

Esta fase se deu início com o término do SSAC. O BDDPM foi avaliado de acordo com as regras do Plano de Análise. Houve concordância entre os 03 avaliadores e o BDDPM recebeu o status de "SUCESSO", pois cumpriu com todos os seus objetivos considerando o desenvolvimento do SSAC.

A Tabela 11 mostra o resumo da análise (MÉTRICAS → PERGUNTAS):

P01: 100,00% de atendimento aos objetivos	**P02: 100% de atendimento aos objetivos**
P03: 100% de atendimento aos objetivos	**P04: 100% de atendimento aos objetivos**
P06: 100% de atendimento aos objetivos	**P08: 80% de atendimento aos objetivos**
P09: 100% de atendimento aos objetivos	**P10: 71,42% de atendimento aos objetivos**
P12: 100% de atendimento aos objetivos	**P13: 100% de atendimento aos objetivos**
P14: 100% de atendimento aos objetivos	**P15: 92,85% de atendimento aos objetivos**
P16: 0% de atendimento aos objetivos	**P17: 25% de atendimento aos objetivos**
P18: 100% de atendimento aos objetivos	**P19: 100% de atendimento aos objetivos**

Tabela 11 – BDDPM: Análise GQM (Bottom-Up) – Métricas → Perguntas

A Tabela 12 mostra o resumo geral da análise (PERGUNTAS → OBJETIVOS) após a consolidação de todos os questionários de avaliação:

O1: 100,00% atendido	**O2: 100,00% atendido**
O3: 87,50% atendido	**O4: 100,00% atendido**

Tabela 12 – BDDPM: Análise GQM (Bottom-Up) – Perguntas → Objetivos

Baseado nos dados analisados, o BDDPM atingiu 100% de seus objetivos no projeto do SSAC.

5.4. Avaliação do Resultado

De acordo com a equipe de avaliação, o BDDPM foi considerado um sucesso. Entretanto, é necessária muita cautela. Existem uma série de questões que devem ser analisadas, pois afetaram diretamente a avaliação. São elas:

1. **Equipe de avaliação formada por profissionais de informática**: esse fator afeta todo os objetivos, mas, em particular, os objetivos O1 e 04 que estão relacionados aos clientes do produto de software. O BDDPM foi criado para que pessoas sem conhecimentos técnicos pudesse utilizá-lo e para que clientes, também sem conhecimentos técnicos, pudesse criar os testes para os produtos dos quais eles são clientes. O fato da equipe de avaliação do BDDPM já ter experiência com computadores e o cliente do SSAC ser um especialista na área, influenciou, positivamente, os resultados medidos.
2. **Equipe de avaliação da PBprev formada por profissionais sem experiência de testes de software com a técnica do BDD**: o projeto do SSAC foi o primeiro projeto da equipe a utilizar o Behavior Driven Development. Nesse sentido, e embora preparações e treinamentos tenham sido ministrados, tudo era novidade. Devido à falta de maturidade

em relação a técnica, essa não seria a melhor equipe para fazer uma avaliação de um produto com enfoque no BDD: a avaliação pode ser distorcida por falsas visões, conceitos, expectativas, empolgação, etc.

3. **O SSAC é um projeto muito pequeno**: embora trate-se de um produto real com o objetivo de ser utilizado dentro de um órgão governamental, o SSAC é um projeto incipiente para se fazer uma análise mais aprofundada e que tivesse requisitos mais exigentes em relação ao BDD. Um resultado de sucesso no SSAC apenas indica que o BDDPM é promissor, ou seja, tem potencial: funcionou em um projeto pequeno, provavelmente funciona em outros projetos pequenos, tem um bom potencial para projetos de médio porte, mas carece de maior avaliação em relação a projeto de grande porte. Sendo assim, o resultado permite fazer conjecturas e prospecções em um cenário otimista.
4. **Quase todos os objetivos terem alcançado 100% de satisfação é estranho**: Foram estabelecidas duas hipóteses para justificar o fato. A primeira delas sugere erro no levantamento das métricas, ou seja, elas não respondem às perguntas de maneira adequada. Esta hipótese foi refutada após nova revisão e avaliação da equipe, que já havia dedicado bastante tempo no levantamento e revisão das mesmas. A segunda hipótese, que foi aceita, sugere que, devido à subjetividade de grande parte das perguntas, seria necessário um maior nível de maturidade em relação ao BDD para responde-las mais adequadamente. A falta dessa maturidade talvez tenha levado a percepção de que tudo estava 100% satisfatório.
5. **Objetivo O3 como elo fraco**: o objetivo 3 foi o que teve menor índice de atendimento, tanto no plano de análise, como na avaliação de fato. Como já foi abordado, por diversas vezes, trata-se, praticamente, do principal objetivo do BDDPM; ou seja, é necessário dar atenção a este problema. Existem dois motivos principais pelo ocorrido, e que merecem atenção: (i) é preciso aumentar o leque de recursos BDD oferecido pela ferramenta, pois o BDDPM só entrega alguns e; (ii) facilitar a adoção da ferramenta envolve disponibilizá-la através de canais de impacto e usar boas estratégias de divulgação na comunidade de teste de software. Até a data da avaliação, a ferramenta só estava disponível para download através de seu repositório GIT.
6. **Primeira avaliação com GQM**: esta foi a primeira avaliação com GQM feita pela equipe, inclusive do próprio fiscal do GQM (time GQM). Isso quer dizer que, em algum(ns) momento(s), pode ter havido falha em algum passo/etapa; o que pode ter prejudicado, em algum nível, a avaliação
7. **Expectativas elevadas da equipe**: o Plano de Análise evidenciou que havia uma expectativa elevada em relação ao BDDPM. Como o BDD estava sendo utilizado pela primeira vez, criou-se a expectativa de que o BDDPM era uma grande solução para o problema. Isso afetou a postura da equipe, mudando-a: deveria ser mais crítica, foi uma postura mais complacente, fazendo com que a avaliação atingisse melhores resultados.

8. **Participação do criador do BDDPM no processo de acompanhamento da avaliação**: A participação de Alvaro Magnum, criador do BDDPM, como acompanhador da avaliação da ferramenta, pode ter impactado na avaliação. Ainda que o avaliador tenha procurado manter uma postura neutra, ele era o chefe da equipe e criou o BDDPM. A equipe pode ter sido influenciada por estes fatores, fazendo com que a avaliação atingisse melhores resultados.

O resultado da avaliação do BDDPM foi resumido em uma nota (pela equipe):

"O BDDPM foi utilizado com sucesso no projeto do SSAC. Seus objetivos foram alcançados e seus resultados levam a acreditar que o projeto é uma boa opção para adoção do BDD em pequenos projetos de desenvolvimento de software. A ferramenta também apresenta grande potencial de crescimento." **Equipe de Avaliação do BDDPM.**

Capítulo 6 – Conclusão

Este é o último capítulo do trabalho. Seu objetivo é falar sobre as dificuldades (Seção 6.1) encontradas no decorrer do projeto de criação de avaliação do BDDPM, falar sobre alguns trabalhos futuros (Seção 6.2) que partiram de necessidades identificadas no decorrer do projeto, elencar algumas contribuições (Seção 6.3) advindas do BDDPM e fazer as considerações finais (Seção 6.4) sobre o trabalho.

Trata-se de um objetivo sucinto e objetivo, dado que grande parte dos comentários e análises foram feitos nos capítulos anteriores.

6.1. Dificuldades Encontradas

Alguns problemas e dificuldades foram encontradas no transcorrer de desenvolvimento deste trabalho, que envolveu o projeto de concepção, criação, desenvolvimento, implantação, testes e avaliação do BDDPM. Foram eles:

- Durante o desenvolvimento do BDDPM, alguns plugins Jquery de interface, ou seja, voltados para a interação homem-máquina, que reduziriam o tempo de desenvolvimento, caso fossem utilizados; tiveram que ser descartados, pois seria necessário pagar uma licença de uso por cada versão distribuída do sistema criado, o que ficaria inviável em relação ao BDDPM que seria distribuído livremente.
- Durante o desenvolvimento do BDDPM, não foi possível testá-lo em outros ambientes que não a plataforma Windows. Em teoria, o plugin deve funcionar em qualquer ambiente com um servidor WEB com suporte a PHP instalado; mas na prática essa teoria não foi avaliada.
- O questionário para levantamento de requisitos do BDDPM obteve menos respostas do que o esperado. Apenas 54 pessoas participaram do processo de avaliação. Esperava-se que este número fosse bem maior, em torno de 80, visto que foi enviado para várias empresas de TI.
- Objetivava-se fazer a avaliação do BDDPM com uma outra técnica, denominada Grounded Theory, mas, por questões de prazos, não foi possível aplica-la. Ela entrou para a listagem de trabalhos futuros.
- A empresa onde o BDDPM foi utilizado, testado e avaliado não é uma empresa cujo foco é a produção de software, ou seja, não possui equipe com experiência em desenvolvimento e teste de software, o que seriam atributos importantes para se avaliar uma ferramenta focada em uma técnica de testes de software: o Behavior Driven Development.
- No início de 2015, momento pelo qual o BDDPM começaria a ser avaliado com a técnica do GQM, o Gerente de TI da empresa, Alvaro Magnum (também criador do BDDPM, integrante da equipe GQM e de acompanhamento da equipe de avaliação do BDDPM), foi exonerado do cargo em comissão que ocupava. Nesse ano houve reestruturação do quadro de servidores devido às mudanças políticas decorrentes das eleições para Governadores de Estado, ocorridas em todo o país no ano

de 2014. Com a mudança da presidência da PBprev, o gerente foi destituído do cargo. Como o projeto já havia sido iniciado, ele continuou, mas o acompanhamento passou a ser a distância.

- Durante a utilização do BDDPM no projeto de desenvolvimento do SSAC, em vez do cliente do projeto, antes de utilizar o BDDPM para fazer a criação dos testes, utilizava o bloco de notas para escrever o código Gherkin. Uma vez com o código em mãos, ele o utilizava para inserir os passos no BDDPM. O cliente justificou que não era necessário fazer isso, mas facilitava o fluxo de pensamento e concepção de testes dele. O objetivo do BDDPM é que os clientes escrevam os testes nele, sem precisar de ferramentas auxiliares. Não se sabe se isso foi um fato isolado, mas foi registrado como problema para estudo e avaliação em outros projetos.
- A equipe de avaliação GQM era pequena e sem experiência de aplicação da técnica em outros projetos, o que pode ter afetado, em algum nível e em algum momento, o processo de avaliação.
- As expectativas da equipe em relação ao BDDPM eram altas, esse fato foi intensificado pelo fato do BDD também ter sido utilizado pela primeira vez pela equipe. As expectativas podem ter impactado o discernimento do time durante o processo de avaliação.
- Não foi possível garantir 100%, após a avaliação, que o BDDPM atende seus objetivos; entretanto, o status de “potencial para atingi-los” foi considerado satisfatório.

6.2. Trabalhos Futuros

Todas as propostas de trabalho a serem executadas no futuro baseiam-se nos 04 objetivos do BDDPM e os resultados de sua avaliação. Os objetivos são:

1. **Fácil de utilizar em termos de usabilidade**
2. **Permitir a aplicação do BDD de maneira eficiente e eficaz**
3. **Facilitar a adoção do BDD**
4. **Permitir a escrita dos testes pelos clientes**

Para impulsionar o **OBJETIVO 1**, os seguintes trabalhos serão executados:

- Adicionar mais assistentes e vídeos de ajuda nas telas do BDDPM.
- Colher dados de formas de utilização da ferramenta e fazer pesquisas em relação à facilidade e usabilidade para implantação de melhorias.

Para impulsionar o **OBJETIVO 2**, os seguintes trabalhos serão executados:

- Adição de mais recursos do BDD através da linguagem Gherkin, com opções como: Background, Transformations, Examples, Data Tables, etc.
- Adicionar integração com o Cucumber.
- Permitir a execução dos testes a partir do próprio BDDPM.

Para impulsionar o **OBJETIVO 3**, os seguintes trabalhos serão executados:

- Aumentar o número de linguagens suportadas: Java, C#, Ruby, Python, etc.
- Aumentar o número de plataformas suportadas: Jira, RedMine, Trac, BugZilla, etc.
- Criar formas de divulgação para impulsionar a promoção da ferramenta no meio acadêmico e em fábricas de software.
- Disponibilizar a ferramenta através de diversos canais.

Para impulsionar o **OBJETIVO 4**, os seguintes trabalhos serão executados:

- Incluir um modo de utilização avançado na interface de criação dos testes para usuários mais experientes que preferirem inserir dados textuais e optarem por este modo.
- Melhorar o Wizard atual para tornar ainda mais intuitiva a criação dos testes para usuários sem conhecimentos técnicos.
- Criar um módulo/extensão/abstração em cima do Gherkin, que é uma linguagem "Business Readable", para torna-la "Business Writable". Objetivo do doutorado.

Uma vez que todos esses trabalhos forem concluídos o BDDPM será novamente analisado. Não apenas com GQM, mas agora também com uma técnica denominada "Grounded Theory"[124], que permite, através de uma análise rigorosa e sistemática, entender determinada situação sem uma teoria a ser testada; apenas através da análise de dados que emergirem de um determinado evento

6.3. Contribuições

Entre as principais contribuições do trabalho, pode-se destacar:

1. **Facilitou a adoção do BDD dentro de uma empresa real** - entregar valor aos clientes faz parte da vantagem competitiva de qualquer empresa. Facilitar a adoção do BDD e permitir que clientes de projetos de desenvolvimento de software participem mais ativamente dos projetos e eles mesmos possam escrever os testes para os produtos que estão comprando, é entregar valor. O ferramental fruto deste trabalho garantiu, dentro de um ambiente real, a PBprev, esta entrega.
2. **Exemplificação real e prática de como aplicar o GQM para avaliação de software** – este documento descreve, em detalhes, como o GQM pode ser aplicado de maneira prática para avaliar um produto de software quanto ao cumprimento de objetivos;
3. **Definição dos requisitos básicos para uma ferramenta BDD no mercado** – após pesquisa bibliográfica e pesquisa com 56 profissionais do mercado de TI, foi possível estabelecer um conjunto de funções mínimas para que uma ferramenta BDD permita, de fato, a utilização e adoção da técnica;
4. **Definição de um conjunto de funcionalidades para facilitar a adoção do BDD no mercado** – dentre eles estão a possibilidade de criação de

[124] http://www.groundedtheory.com

testes pelo próprio cliente do produto de software que pode "escrever" os testes de maneira remota e através de um Wizard; e a possibilidade de acompanhamento de testes em tempo real pelos stakeholders;

5. **Motor de conversão de código de alto nível, descrito em Gherkin, para código de teste em C#** – uma grande vantagem do BDDPM é permitir a criação de códigos de teste, escritos em uma linguagem de programação, serem gerados a partir da definição do comportamento do software feito pelo cliente. Este motor utiliza expressões regulares e é de código aberto, podendo ser adaptado e reutilizado para geração de código de teste em outras linguagens como Java, Python, Ruby, PHP, etc.
6. **Criação de uma ferramenta open-source** – a opção de tornar o BDDPM uma ferramenta de código aberto, livre para ser incrementado e melhorado pela comunidade e profissionais de TI ligados ao desenvolvimento e testes de software, permitirá, ainda mais, facilitar a adoção do BDD e, consequentemente, ir ao encontra de um ambiente de desenvolvimento orientado por comportamento.
7. **Primeiro plugin BDD para o Mantis** – o repositório oficial do Mantis não listava opções para trabalho com o Behavior Driven Development. O BDDPM é o primeiro plugin a oferecer este recurso para a plataforma.
8. **Tratamento de tema de interesse direto da empresa em que o mestrando trabalha** – O BDDPM melhorou e facilitou a atividade de teste de software dentro da Paraíba Previdência e, consequentemente, melhorou os produtos de softwares criados dentro da instituição. Este objetivo está alinhado aos objetivos do mestrado profissional que, conforme já citado anteriormente, diz o seguinte: "*O MPROF difere do Mestrado Acadêmico por **lidar com problemas que o profissional vivencia dentro do mercado**. Sendo assim, **os projetos de conclusão normalmente tratam de temas de interesse direto da empresa em que o mestrando trabalha**. A interação dos acadêmicos com as problemáticas enfrentadas no dia-a-dia das instituições **contribui para a melhoria das empresas**, e consequentemente para o desenvolvimento da região*"[125].

6.4. Considerações Finais

O BDDPM foi criado com alguns objetivos iniciais, o principal deles: ser uma ferramenta capaz de agregar recursos que facilitem a adoção do BDD no mercado. Durante os quase 06 (meses) de desenvolvimento do projeto, objetivos foram refinados, recursos do plugin foram adicionados e excluídos, problemas foram detectados, etc.; entretanto, a partir do momento em que o BDDPM passou a ser utilizado dentro de um projeto real, percebeu-se que ele agregou valor, principalmente para uma equipe não familiarizada com a metodologia de testar comportamento de software. Todos ficaram satisfeitos com os resultados alcançados.

[125] http://www2.cin.ufpe.br/site/secao.php?s=3&c=116

A aplicação de uma técnica de avaliação/medição reconhecida (GQM) consolidou o pensamento de que o BDDPM é um plugin que pode ser evoluído para que seus objetivos sejam alcançados, de fato, em outras organizações.

Independentemente de conjecturas, o BDDPM foi adotado pela PBprev que irá utilizá-lo nos próximos projetos de desenvolvimento de software do órgão. Há de se considerar que o BDD foi aplicado pela primeira vez, e com sucesso, por uma equipe sem experiência com a técnica. Estes resultados vão ao encontro dos objetivos do BDDPM. Sendo assim, e após análise da avaliação da equipe GQM, este trabalho conclui que o BDDPM tem grande potencial de atingir seus objetivos, principalmente em projetos de pequeno e médio porte.

Referências

Arnold, D., Corriveau, J., Shi, W.: Validation against Actual Behavior: Still a Challenge for Testing Tools. Software Engineering Research and Practice (SERP). CSREA Press. (2010).

Basili V.R, "Data Collection, Validation, and Analysis," in Tutorial on Models and Metrics for Software Management and Engineering, IEEE Catalog No. EHO-167-7, 1981, pp. 310-313.

Basili V.R, "Software Modeling and Measurement: The Goal Question Metric Paradigm," Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.

Basili V.R, G. Caldiera and H. D. Rombach. Goal Question Metric Paradigm. In John C. Marciniak, editor, Encyclopedia of Software Engineering, volume 1. John Wiley & Sons, 1994.

Basili V.R and D. M. Weiss. A Methodology for Collecting Valid Software Engineering Data. IEEE

Basili, V.R.; J. Heidrich, M. Lindvall, J. Münch, C.B. Seaman, M. Regardie, A. Trendowicz (2009). "Determining the impact of business strategies using principles from goal-oriented measurement". Business Services: Konzepte, Technologien, Anwendungen. Internationale Tagung Wirtschaftsinformatik. Books OCG. Vienna, Austria: Österreichische Computer Gesellschaft. ISBN 978-3-85403-246-5.

Beck, K. Test Driven Development: By Example. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN: 0321146530.

Beck K., M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland e D. Thomas. Manifesto for Agile Software Development. Acessado em Novembro de 2014. URL: http://www.agilemanifesto.org.

Birk Andreas, Rini van Solingen, Janne Järvinen, ‘Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-depth, Long-term Investigation at Schlumberger RPS’, Proceedings of the 5th International Symposium on Software Metrics (Metrics’98), Bethesda Maryland, November 19-21, 1998.

Carvalho R., R. Soares Manhães, and F.L. de Carvalho, Filling the Gap between Business Process Modeling and Behavior Driven Development, CoRR, 2008.

Chelimsky, D. The Rspec Book: Behaviour Driven Development With Rspec, Cucumber, and Friends. Pragmatic Bookshelf Series. Pragmatic Programmers, LLC, 2010. ISBN: 9781934356371.

Christiane G. von Wangenheim, Günther Ruhe. Análise de Custo e Benefício de Mensuração Baseada em GQM.X CITS QS, 1999

Cucumber. Acessado em Julho de 2014. URL: http://cukes.info.

Daskalantonakis M.K. A Practical View of Software Measurement and Implementation Experiences within Motorola. IEEE Transactions on Software Engineering, Vol. 18, No. 11, 1992.

Differding C., B. Hoisl, C. Lott. Technology Package for the Goal/Question Metric Paradigm. Technical Report 281-96, University of Kaiserslautern, Department of Computer Science, D-67653 Kaiserslautern, Germany, 1995.

Feduke C. Instant RSpec Test - Driven Development How-to. Packt Publishing, 2013. ISBN: 1782165223, 9781782165224.

Generative Programming. Acessado em Julho de 2014. URL: http://www.programtransformation.org/Transform/GenerativeProgrammingWiki.

Glaser, Barney. Strauss, Anselm. Awareness of Dying. Chicago: Aldine Publishing Company. 1965. ISBN 0-202-30763-8.

Hellesoy A. e Wynne M. The Cucumber Book: Behaviour-Driven Development for Testers and Developers. Pragmatic Programmers. Pragmatic Bookshelf, 2012. ISBN: 9781934356807.

Hoisl B., M. Oivo, G. Ruhe, and F. van Latum. No Improvement without Feedback: Experiences from goal-oriented measurement at Schlumberger. Fifth European Workshop on Software Process Technology, Nancy, France, October 1996.

Is TDD Dead? Acessado em Julho de 2014. URL: http://martinfowler.com/articles/is-tdd-dead.

JBehave. Acessado em Julho de 2014. URL: http://jbehave.org.

Jerkovic, John I. SEO Warrior. O'Reilly, 2009.ISBN: 978-0-596-15707-4.

Krzysztof Czarnecki, Generative Programming - Principles and Techniques of Software Engineering Based on Automated Configuration and Fragment-Based Component Models, 1998, Department of Computer Science and Automation.

McCracken, D.D., Digital Computer Programming, John Wiley, 1957.

McFarlan, F.W. 1984. Information Technology Changes the Way You Compete, Harvard Business Review, 62(3): 98-103.

Nagappan, N., Maximilien, E., Bhat, T., Williams, L.: Realizing Quality Improvement through Test Driven Development: Results and Experiences of Four Industrial Teams. Empirical Software Engineering, pp. 289-302 (2008).

Naik, Kshirasagar, Tripathy, Priyadarshi. Software testing and quality assurance. Wiley. 2008. ISBN 978-0-471-78911-6

Necas I. “BDD as a Specification and QA Instrument”. Diplomová práce. Masarykova univerzita, Fakulta informatiky, 2011 [cit. 2014-01-30].

North D. Introducing BDD. Acessado em Julho de 2014. URL: http://dannorth.net/introducing-bdd.

Popular Bug Tracking Software. Acessado em Julho de 2014. URL: http://www.softwaretestinghelp.com/popular-bug-tracking-software.

Rediscovery of Test Driven: https://www.quora.com/Why-does-Kent-Beck-refer-to-the-rediscovery-of-test-driven-development

Ress, Ana Paula , Renato y Salerno, Mário Sérgio. Test-Driven Development as an Innovation Value Chain. Journal of Technology Management & Innovation, 2013, vol.8, no., p.10-10. ISSN 0718-2724.

Rogerio R. S. M. Atem de Carvalho Fernando Luiz de Carvalho e Silva. “Business Language Driven Development: Joining Business Process Models to Automated Tests”. CoRR (2011).

Rombach H.D. Practical Benefits of Goal-Oriented Measurement. Software Reliability and Metrics, Elsevier Applied Science, 1991.

RSpec. Acessado em Julho de 2014. URL: http://rspec.info.

Selenium. Acessado em Dezembro de 2014. URL: http://www.seleniumhq.org.

Soingen, R.V., Berghout, E.: The Goal/Question/Metric Method: a practical guide for quality improvement of software development. McGraw-Hill Co. (1999).

Solingen R., F. van Latum, M. Oivo, and E. Berghout. Application of Software Measurement at Schlumberger RPS: Towards Enhancing GQM. Proceedings of the Sixth European Software Cost Modelling Conference (ESCOM), May 1995.

Solingen R. Goal-Oriented Software Measurement in Practice: Introducing Software Measurement in Schlumberger Retail Petroleum Systems. Master Thesis Report, Schlumberger RPS, Netherlands, 1995.

Solingen, R, E.W. Berghout, E. Kooiman, 'Assessing feedback of measurement data: Schlumberger practices with reflection to theory', Proceedings of the 4th International Symposium on Software Metrics (Metrics'97).

Solingen R., Rini; Egon Berghout (1999). The Goal/Question/Metric Method. McGraw-Hill Education. ISBN 0-07-709553-7.

Solis, C and Xiaofeng Wang (2011). A Study of the Characteristics of Behaviour Driven Development. In the proceedings of the 37th EUROMICRO conference on Software Engineering and Advanced Applications (SEAA) (pp. 383 - 387). Oulu, Finland.

Sommerville, Ian. Software Engineering. 9. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1.

SpecFlow. Acessado em Julho de 2014. URL: http://www.specflow.org.

Tavares H.L, G. G. Rezende, V. M. dos Santos, R. S. Manhães e R. A. de Carvalho. "A tool stack for implementing Behaviour-Driven Development in Python Language". CoRR abs/1007.1722 (2010).

The ROI of TDD: http://powersoftwo.agileinstitute.com/2012/08/the-roi-of-test-driven-development.html

VAGAS. Você é Júnior, Pleno ou Sênior? Acessado em Janeiro de 2015. URL: http://www.vagas.com.br/profissoes/acontece/no-mercado/voce-junior-pleno-senior.

Wangenheim, Christiane Gresse von. Utilização do GQM no desenvolvimento de software. Laboratório de Qualidade de Software, Instituto de Informática, Universidade do Vale do Rio dos Sinos. São Leopoldo, 2000.

Wayne, YE. Instant Cucumber BDD How-to. Packt Publishing. 2013. ISBN 978-1-78216-348-0.

What is Grounded Theory? Acessado em Dezembro de 2014. URL: http://www.groundedtheory.com/what-is-gt.aspx.

You won't believe how old TDD is. Acessado em Dezembro de 2014. URL: https://arialdomartini.wordpress.com/2012/07/20/you-wont-believe-how-old-tdd-is.

Apêndice 1 – Questionário para Criação da do BDDPM

1. **Em sua opinião, para uma aplicação WEB ser considerada de fácil uso e simples, inclusive para usuários com pouco, ou nenhum, conhecimento técnico; qual deveria ser o número de telas úteis dessa aplicação? Considere que TELA ÚTIL é aquela que contém ou permite a entrada informações importantes e/ou essenciais para utilização do sistema.**
    a. De 1 a 4
    b. De 4 a 8
    c. De 8 a 12
    d. Mais de 12
    e. Outro
2. **Qual seu nível de experiência/conhecimento em relação ao BDD (Behavior Driven Development)?**
    a. Desconheço completamente
    b. Sei o que é, mas nunca trabalhei com a técnica
    c. Menos de 1 ano de experiência
    d. Menos de 3 anos de experiência
    e. Mais de 3 anos de experiência
3. **Em sua opinião, considerando uma aplicação WEB voltada para o BDD (Behavior Driven Development) e que permite a criação de testes de aceitação com Gherkin, qual a melhor proposta em termos de facilidade de uso? (Inclusive para usuários com pouco, ou nenhum, conhecimento técnico).**
    a. Campo de texto livre para inserção do Gherkin
    b. Recursos de arrastar e soltar componentes para “montagem” do Gherkin
    c. Mistura entre campos de múltipla escolha (combo box, radio, etc.) e campos de texto livre para “montagem” do Gherkin
    d. Importação de testes de aceitação prontos de outra ferramenta
    e. Nenhuma das opções e/ou não me considero apto a responder à pergunta
4. **Na sua opinião, qual seria o recurso mais importante em uma aplicação WEB voltada para o BDD (Behavior Driven Development)?**
    a. Possibilidade de acompanhamento do resultado dos testes
    b. Execução automática dos testes de aceitação
    c. Geração de código de testes
    d. “Eu” poder escrever testes de aceitação de maneira fácil
    e. Outro
5. **Na sua opinião, qual seria o recurso mais importante PARA O CLIENTE DE UM PRODUTO DE SOFTWARE em uma aplicação WEB voltada para o BDD (Behavior Driven Development)?**
    a. Possibilidade de acompanhamento do resultado dos testes
    b. Execução automática dos testes de aceitação
    c. Geração de código de testes
    d. “Eu” poder escrever testes de aceitação de maneira fácil
    e. Outro
6. **Você é:**
    a. Um(a) desenvolvedor(a)/testador(a) de software
    b. Um(a) gestor(a) de projetos de software
    c. Um(a) cliente de projeto de software
    d. Diretor(a)/Sócio(a) de uma fábrica de software
    e. Outro

Apêndice 2 – Questionário para Avaliação GQM

1. Em relação a facilidade de uso do BDDPM, que nota, de 1 a 4, você atribui à ferramenta? Considere uma escala onde 1 significa "Difícil de Usar" e 4 "Fácil de Usar". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fácil" e "Difícil", respectivamente.
2. Para utilizar o BDDPM de maneira satisfatória, qual o nível (de 1 a 4) de conhecimento técnico empregado por você? Considere uma escala onde 1 significa "Nenhum Conhecimento Técnico" e 4 significa "Muito Conhecimento Técnico". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Nenhum" e "Muito", respectivamente.
3. Considerando o objetivo do BDDPM de permitir a utilização da técnica do BDD em um projeto de desenvolvimento de software, bem como os recursos oferecidos pela ferramenta; classifique a interface da ferramenta atribuindo uma nota de 1 a 4. Considere uma escala onde 1 significa "Pouco Intuitiva" e 4 significa "Muito Intuitiva". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.
4. Considerando os recursos do BDDPM, como você classificaria sua interface considerando-a como um meio de acesso eficaz e atraente a estes recursos? Considere uma escala onde 1 significa "Baixa Qualidade" e 4 significa "Alta Qualidade". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Baixo" e "Alto", respectivamente.
5. Qual a média de tempo (em minutos) para execução da fase de testes dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)
6. Qual a média de tempo (em minutos) para execução da fase de testes dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?
7. Qual a média de tempo (em minutos) de duração dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)
8. Qual a média de tempo (em minutos) de duração dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?
9. Levando-se em consideração a utilização do BDDPM no projeto de desenvolvimento de software, como você classifica (nota de 1 a 4) o impacto dos benefícios agregados ao projeto? Considere uma escala onde 1 significa "Baixo Impacto" e 4 significa "Alto Impacto". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Baixo" e "Alto", respectivamente.
10. Considerando a técnica do BDD, como você classificaria (nota de 1 a 4) o nível de alinhamento e atendimento do BDDPM a ela? Considere uma escala onde 1 significa "Fraco" e 4 significa "Forte". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fraco" e "Forte", respectivamente.
11. Qual o quantitativo médio de falhas/bugs encontrados nos produtos dos projetos de desenvolvimento de software da empresa, sem a utilização do BDDPM? (Pergunta de Suporte)
12. Qual o quantitativo médio de falhas/bugs encontrados nos produtos dos projetos de desenvolvimento de software da empresa, com a utilização do BDDPM?
13. Qual o seu nível (nota de 1 a 4) de satisfação em relação a eficácia e utilização do BDDPM em um projeto de desenvolvimento de software? Considere uma escala onde 1 significa "Pouco Satisfeito" e 4 significa "Muito Satisfeito". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.
14. Ao utilizar o BDDPM no projeto de desenvolvimento de software, você precisou recorrer à outras ferramentas para aplicação do BDD? Responda com Sim, ou Não.
15. Você pretende utilizar o BDDPM em outros projetos? Responda com Sim, ou Não.
16. Considerando a forma de disponibilização do BDDPM, como você classificaria (nota de 1 a 4) sua acessibilidade e visibilidade? Considere uma escala onde 1 significa "Pouca acessibilidade / visibilidade" e 4 significa "Muita acessibilidade / visibilidade". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Pouco" e "Muito", respectivamente.
17. Quantas linguagens de programação devem ser suportadas, em sua opinião, para que BDDPM tenha grande aceitação no mercado?
18. Qual o seu nível de dificuldade (nota de 1 a 4) para criar um teste no BDDPM? Considere uma escala onde 1 significa "Fácil" e 4 significa "Difícil". As notas 2 e 3 representam níveis intermediários com aproximação dos conceitos de "Fácil" e "Difícil", respectivamente.
19. Quanto tempo, em média (minutos), você leva para criar um teste no BDDPM?

Apêndice 3 – Autorização de Utilização do BDDPM na PBprev e Divulgação dos Resultados

PBprev
PARAÍBA PREVIDÊNCIA

AUTORIZAÇÃO

Eu, Alvaro Magnum Barbosa Neto, gestor de TI da Paraíba Previdência, matrícula 460.212-9, autorizo a utilização do BDDPM (plugin para aplicação da técnica de teste de software denominada "Behavior Driven Development") no projeto SISPROTO – SSAC desta instituição, para fins de avaliação acadêmica (dissertação de mestrado de aluno da UFPE – Universidade Federal de Pernambuco).

A partir desta data, ficam disponíveis para o projeto, 04 colaboradores e toda infraestrutura de TI da PBprev para serem utilizados de acordo com as necessidades de utilização e avaliação do BDDPM até a data de finalização do projeto.

Ressalte-se que, para todos os fins e sem ressalvas, que o SISPROTO – SSAC é de propriedade da PBprev e foi utilizado apenas como suporte para avaliação do BDDPM. A presidência da PBprev não autoriza detalhamento de códigos da ferramenta, apenas a divulgação do nome da empresa, da avaliação dos resultados e da metodologia empregada.

João Pessoa - PB, 01 de dezembro de 2014

Alvaro Magnum B. Neto

_____

ALVARO MAGNUM BARBOSA NETO

GESTOR DE TI – PARAÍBA PREVIDÊNCIA

MATRÍCULA 460.212-9

TELEFONE: (0xx83) 2107.1128

E-MAIL: alvaromagnum@pbprev.pb.gov.br

Apêndice 4 – Relato Pessoal na Área de Teste de Software

Comecei a trabalhar na área de Desenvolvimento de Software aos 18 anos, atualmente tenho 30. Durante esse período tive a oportunidade de ocupar cargos de nível júnior, pleno e sênior e trabalhar em empresas de diversos portes, desde as que operavam "no vermelho", passando pelas que faturam alguns milhares, até as que movimentam bilhões de reais por ano. Também tive a oportunidade de trabalhar em projetos locais de pequeno porte e de grandes projetos distribuídos globalmente. Tive, ainda, a oportunidade de vivenciar duas realidades bem distintas uma vez que trabalhei em empresas privadas e atualmente trabalho em uma instituição pública. Fui estagiário, técnico, consultor e gerente. Passei por empresas de pequeno, médio e grande porte, com setores financeiros que movimentavam mais de R$ 115 mi/mês.

Percebi que cada uma dessas empresas se preocupava bastante com “construir” um produto de alta qualidade e livre de defeitos, porém nem todas davam a mesma atenção para “construir” o que o cliente queria, exatamente com o comportamento previsto: o velho e clássico problema do “V & V”, verificação e validação. Havia, nas empresas, grande preocupação com a verificação, ou seja, checar se o produto atendia aos seus requisitos funcionais e não funcionais; porém não com a validação, isto é, checar se o produto realmente estava de acordo com as expectativas do cliente. Nem sempre um produto que funciona é o que o cliente realmente quer. Um produto livre de erros pode ser completamente inútil para um cliente se não este não se comporta da maneira como ele espera.

Nesse sentido todos adotavam “testes de unidade” com a abordagem do TDD como requisito básico e primordial para testes. Cada desenvolvedor tinha que “escrever” seu código com sua bateria mínima de testes de unidade. Depois o software passava por uma bateria de testes exaustivos junto aos testadores cujo os objetivos eram de encontrar defeitos. Muitos testes eram realizados para evitar que um único erro passasse despercebido, mas não havia teste para saber se a funcionalidade satisfazia a necessidade cliente.

Uma possível explicação para as coisas serem assim pode ser o fato de que, muitas vezes, manter o contato com o cliente é custoso. Nem sempre o cliente está disponível para acompanhar o desenvolvimento de seu produto, o cliente pode até mesmo estar localizado distante geograficamente; em um outro país, por exemplo. Muitas vezes os requisitos são levantados em uma única reunião e outros contatos são realizados por telefone ou pela Internet. Muitas vezes esses encontros só são realizados quando existe algo palpável que pode ser mostrado e entregue, e o cliente não tem acesso ao produto durante o desenvolvimento das entregas.

Sob esta perspectiva, era muito fácil descrever e escrever funções técnicas. Por isso o TDD foi tão utilizado: desenvolvedores tinham acesso às funções e escreviam código e testes para estas funções. O comportamento do software era esquecido, ou, um melhor atributo, evitado. Evitado por ser custoso. Ter acesso ao cliente tempo o suficiente ou com a frequência suficiente para descrever, testar e acompanhar comportamentos poderia ser muito custoso.

BDD é uma evolução do TDD, continua sendo TDD e agrega o teste de comportamento; entretanto, em 12 anos de experiência profissional nunca participei de um projeto onde BDD fosse uma exigência assim como o TDD. Nunca me pediram BDD, nunca pedi BDD. Por quê? O que faltou? Será que alguma técnica ou estratégia de adoção do BDD poderia ser utilizada da mesma maneira em todas as empresas, ou teria que se adaptar a cada uma delas?

Estas perguntas me levaram a algumas reflexões em relação ao BDD e foram alguns dos motivos que me fizeram pensar neste trabalho.