FACULDADE DE TECNOLOGIA DA ZONA LESTE

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

BRUNO HARNIK	**RA: 1110481823052**
FERNANDA REIS	**RA: 1110481823022**
LUIZ FERNANDO	**RA: 1110481823051**
RAQUEL MARTINS	**RA: 1110481823032**
ROBSON FERREIRA	**RA: 1110481823026**

BEHAVIOR DRIVEN DEVELOPMENT (BDD)

DISCIPLINA: ENGENHARIA DE SOFTWARE I

PROFESSORA: CRISTINA CORRÊA DE OLIVEIRA

SÃO PAULO

2019

Sumário

INTRODUÇÃO

Em tecnologia da informação costumamos dizer que “os profissionais da tecnologia não têm muita criatividade” e, de certa forma, podemos confirmar isso após alguns meses de integração. Com BDD não é diferente.

Behavior Driven Development, ou desenvolvimento guiado por comportamento, é autoexplicativo. Ipsis litteris.

O termo - e, consequentemente, o processo - foi criado em 2003 pelo especialista em mudanças tecnológicas e organizacionais Dan North, a partir de outra metodologia, dirigida a partir de testes (TDD).

Quando falamos de desenvolvimento de software, uma série de componentes se faz necessária: precisamos levantar e analisar requisitos, projetar, implementar, testar e implantar. Mas como? Quem são os atores de todo esse processo?

Os *stakeholders* são todos os agentes interessados na proposta e que participam de alguma maneira do desenvolvimento, seja com a ideia, com o desenvolvimento, com o teste, ou a documentação. Quanto mais stakeholders, maior é a probabilidade de ocorrer alguma falha na iteração (ou até mesmo a ausência dela) e o software pode não atingir à expectativa do cliente. Isso é mais provável de acontecer em equipes que se utilizam do TDD, onde um analista de negócios dirige a informação aos desenvolvedores, documentadores e *testers* e estes executam as etapas individualmente.

TDD – TEST DRIVEN DEVELOPMENT

TDD é a forma de desenvolvimento ágil que se baseia em criar e executar testes automáticos antes mesmo de o programa ter sido criado. Assim, a lógica é desenvolvida de forma simplista no início e poderá ser utilizada como referência para a criação do programa de maneira segura e bem controlada. Para a utilização do TDD segue-se três passos primordiais, chamados de “vermelho”, em que um teste que não funciona é escrito; “verde”, em que o teste antes escrito é trabalhado em código para que funcione; e “refatoração”, etapa em que há a eliminação de todas as duplicações criadas e a organização do código (BECK, 2003)

Ou seja, para o TDD o mais importante é desenvolvimento ágil do código em si, não havendo muitas iterações com os stakeholders, como mostrado na imagem abaixo.

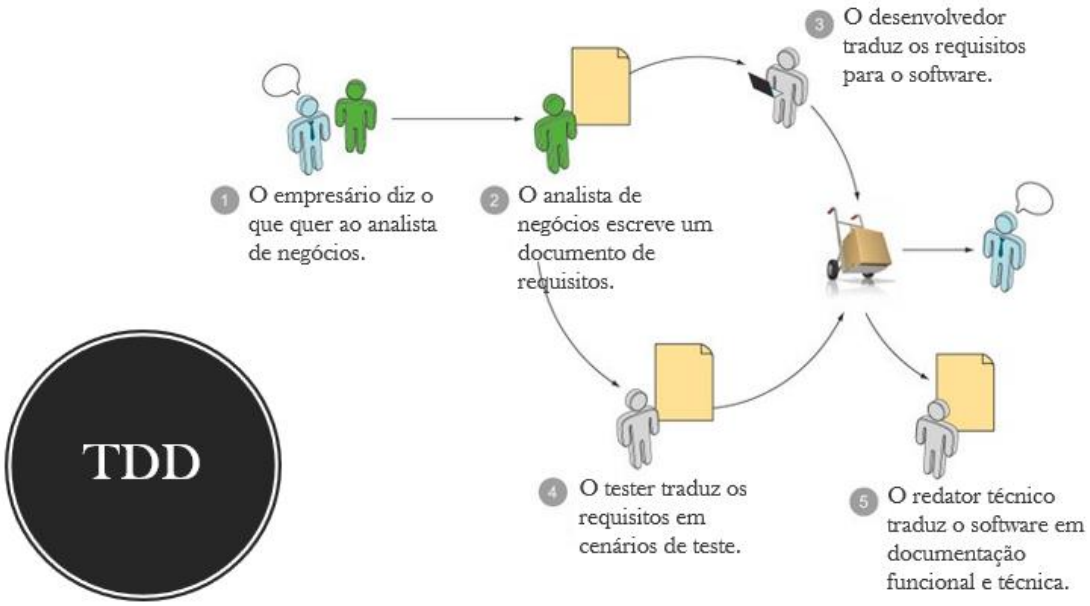

Figura 1 - esquema de TDD

Em alguns sistemas específicos, como por exemplo onde não há possibilidade de comunicação entre todos os *stakeholders*, o TDD se mostra mais viável, mas quando precisamos analisar o comportamento, a razão pela qual o software foi criado, utilizamos o BDD.

BDD – BEHAVIOR DRIVEN DEVELOPMENT

Dan North acredita que o foco deve ser na comunicação entre todas as pessoas envolvidas no projeto; técnicos e não técnicos. Em consequência, sua linguagem, Gherkin, permite a comunicação efetiva diante de sua ubiquidade.

O grande propósito do BDD é a entrega de valor para o negócio. Ele define e entende o comportamento com foco no valor e no usuário. O que importa é o comportamento do software, guiado pela iteração - consulta periódica ao cliente na fase de desenvolvimento - amiúde a fim de entregar o essencial, nem mais nem menos. A dinâmica durante o desenvolvimento é mostrada na imagem abaixo.

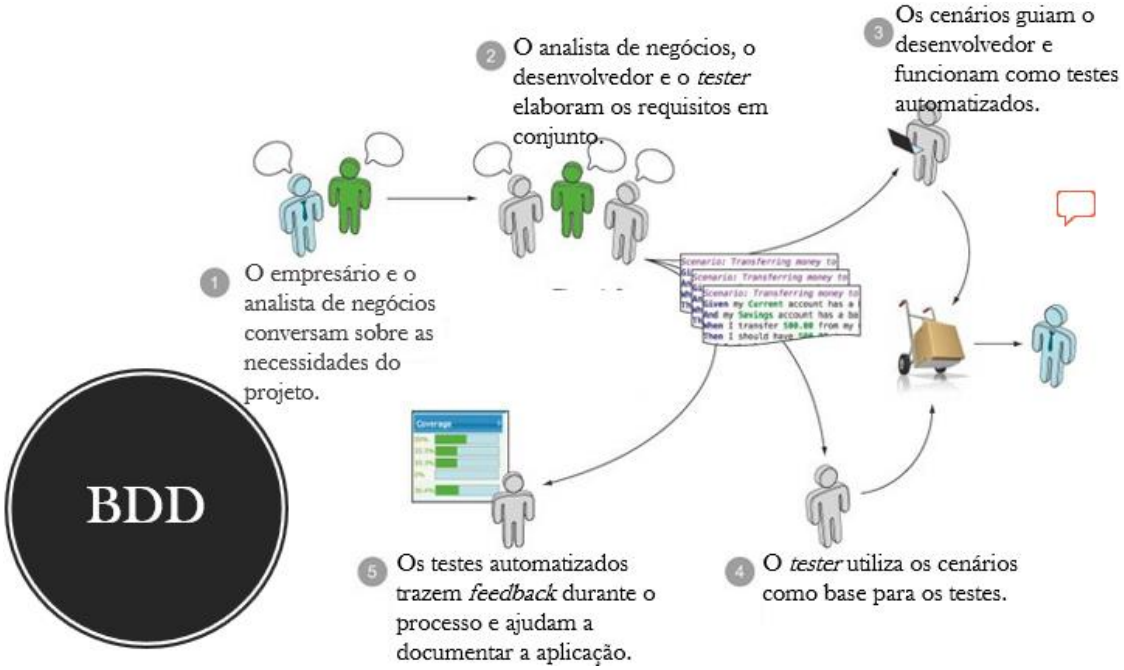

Figura 2 - BDD.

Podemos afirmar que o BDD é uma metodologia:

- outside-in → de fora para dentro, com visão de valor de negócio;
- pull-based → focado, autossuficiente;
- multiple-stakeholder → múltiplos stakeholders, colaborativa, centrada no usuário e todos os que se importam com a qualidade do produto final;
- multiple-scale → vários níveis de aplicação e de códigos;
- high automation → automatizada, ágil, iterativa, de linguagem ubíqua;

LINGUAGEM GHERKIN

"Se pudéssemos desenvolver um vocabulário consistente para analistas, testadores, desenvolvedores e pessoas da área de negócios, então estaríamos a caminho de eliminar algumas das ambiguidades e falhas de comunicação que ocorrem quando pessoas da área de tecnologia falam com pessoas da área de negócios." - Dan North

Para a aplicação do BDD, histórias principais devem ser criadas. Histórias são exemplos de funcionalidades do programa em desenvolvimento. São casos elaborados para mostrar como

será o comportamento das pessoas e do programa desenvolvido naquela circunstância ou em semelhantes. Utilizaremos como exemplo de narrativa um saque em um caixa eletrônico.
A história, em Gherkin, possui 3 componentes: "como", "eu quero" e "para". O "como" refere-se ao "eu", como usuário do serviço ou produto. "Eu quero" remete à execução do usuário, o que ele quer fazer. "Para" é a finalidade da execução.

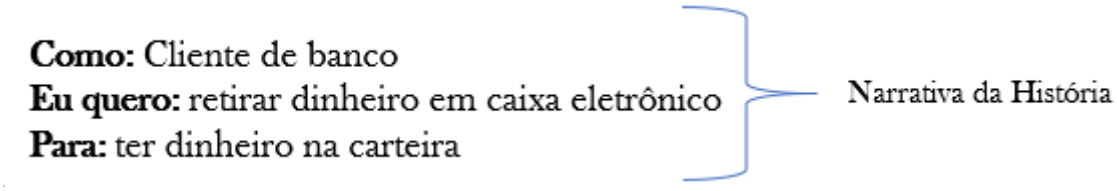

Figura 3- Exemplo de história.

A partir de uma história principal, criam-se cenários: situações específicas, identificadas como condições. São compostos por "dado que" (given), "quando" (when) e "então" (then).
O propósito do "Dado" é colocar o sistema em um estado conhecido antes que o usuário comece a interagir com o mesmo. "Quando" descreve qualquer ação de interação do usuário com o sistema. O "Então" visa mostrar as saídas, os resultados das ações executadas.
No exemplo, utilizamos o cenário "conta com saldo positivo".

CENÁRIO 1 – Conta com saldo positivo

Dado que: a conta está com R$ 500,00 de saldo
Quando: for solicitado um saque de R$ 400,00
Então: receber o valor de R$ 400,00
E: a conta deve ficar com saldo em R$ 100,00

CENÁRIO 2 – Conta com saldo negativo

Dado que: a conta está com R$ 100,00 de saldo negativo
Quando: for solicitado um saque de R$ 400,00
E: não possuir valor de cheque especial disponível
Então: receber mensagem de rejeição de operação
E: não receber valor

Figura 4 - Exemplos de cenários

FRAMEWORK JBEHAVE

O JBehave foi o primeiro framework de BDD, desenvolvido pelo próprio Dan North. Ele define um objeto modelo que permite mapearmos diretamente os fragmentos de cenários para classes Java.

Os frameworks de BDD são desenvolvidos para tornar as práticas de desenvolvimento mais acessíveis e intuitivas tanto para especialistas quanto para novatos. Seu vocabulário é baseado no comportamento e seus testes, automatizados.

BIBLIOGRAFIA

BECK, Kent. Test-Driven Develpment: by exemple. Addison-Wesley - Boston, 2003.
Dan north & Associates (dannorth.net em Abril de 2019)
JBehave (jbehave.org em Abril de 2019)