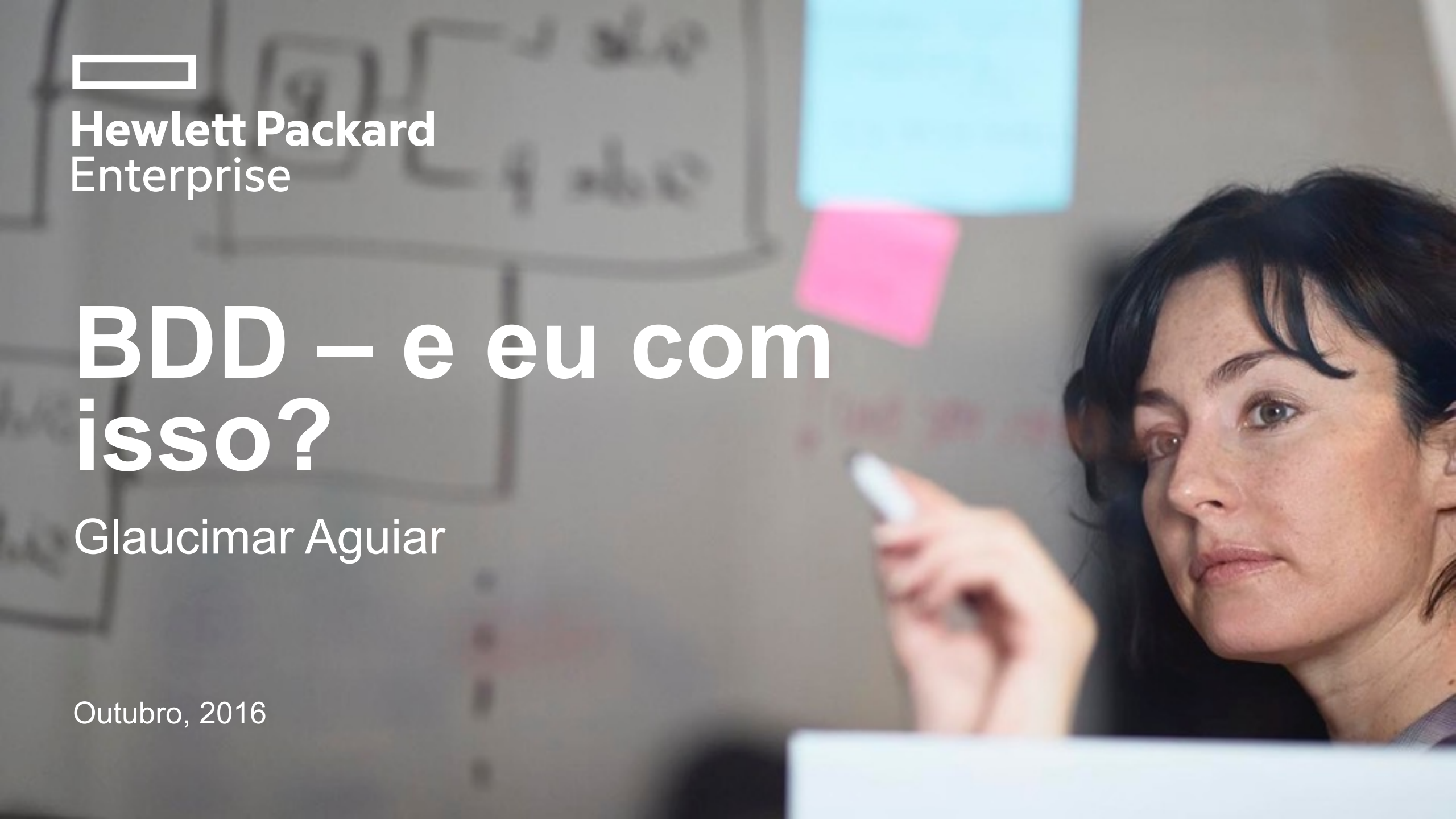


Hewlett Packard
Enterprise

BDD – e eu com isso?

Glaucimar Aguiar

Outubro, 2016

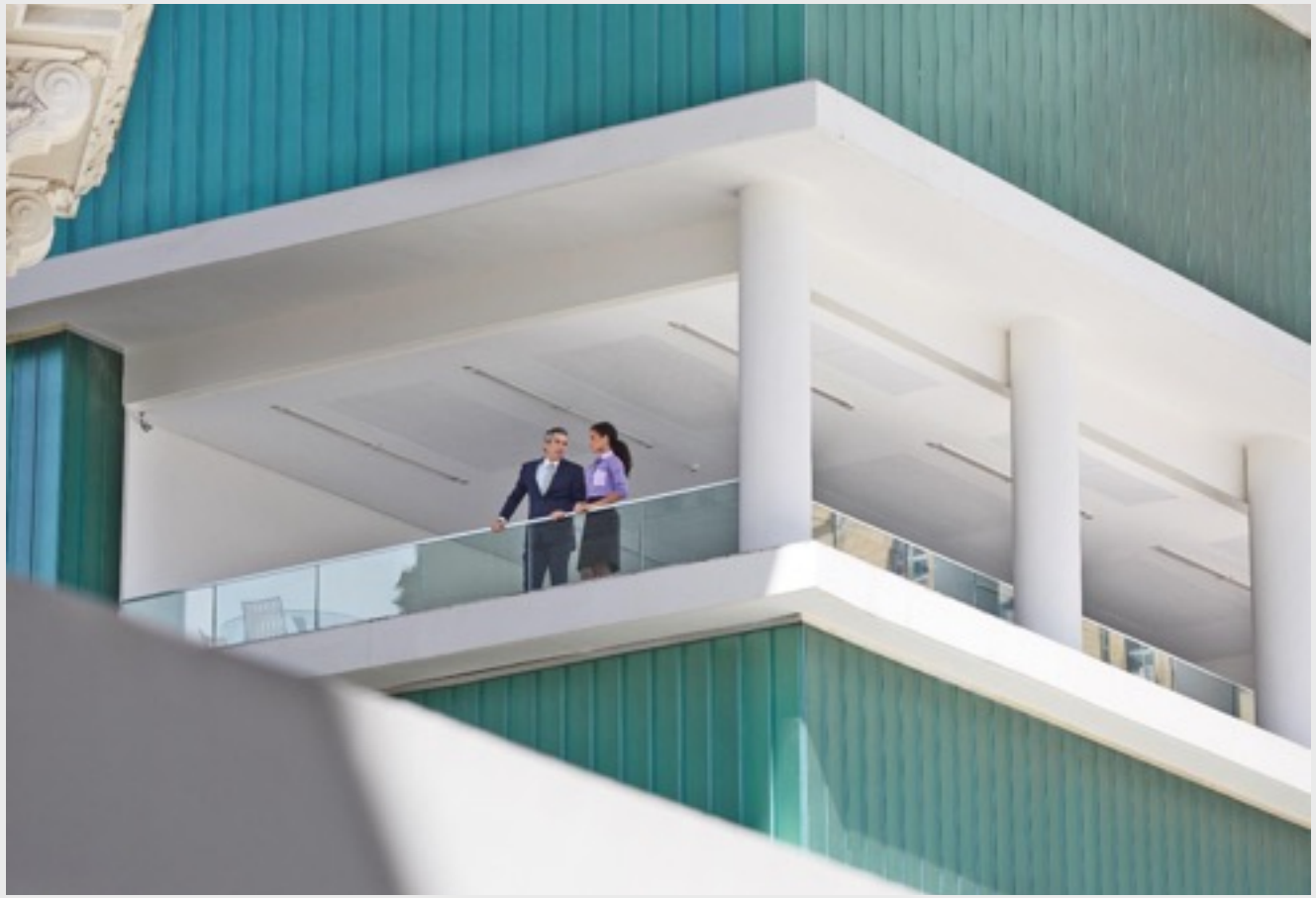
Quem sou...

E o que esperar desta conversa

Sobre desenvolvimento de software...

Desafios em projetos de desenvolvimento de software

- Projetos atrasam ou excedem **muito** o orçamento inicial
- Entregam software que não atende às necessidades dos usuários
- Aplicações com muitos problemas em produção
- O código é de difícil manutenção
- A comunicação é difícil!
- Outros fatores:
  - Corrigir defeitos é caro
  - Especialmente em estágios avançados do desenvolvimento…



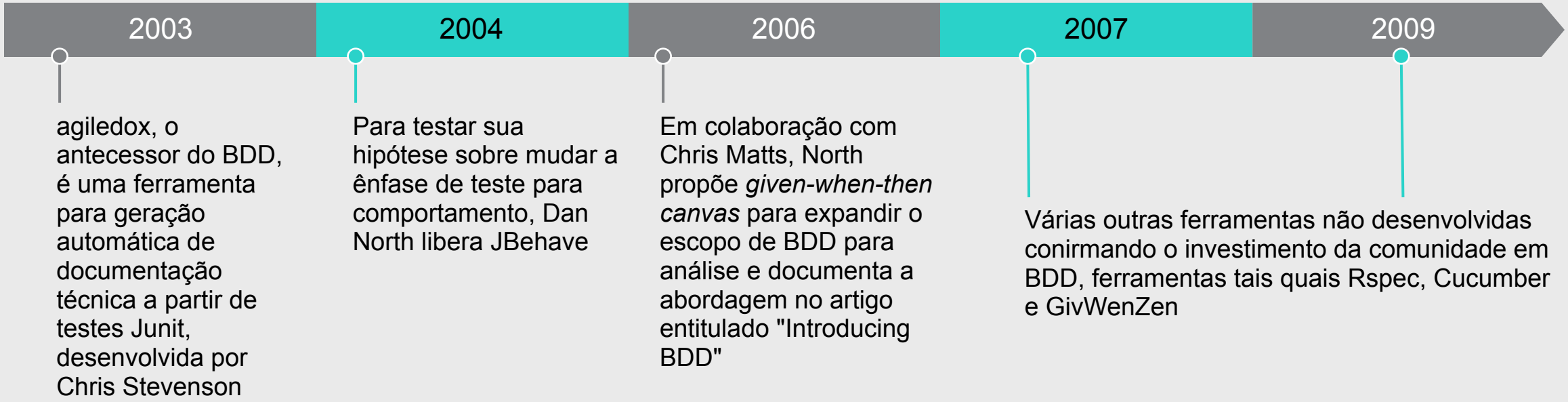
Se pudéssemos entregar melhor...

– Funcionalidades (ou valor!) ao invés de módulos, componentes, classes…
– Re-priorizar com frequência
– Focar principalmente nas funcionalidades de alto valor agregado
– Diminuir ou estabilizar o custo da mudança
– Adaptar-se a nova realidade
– Encontrar os problemas antes
– Enfim.. aprender enquanto desenvolvemos…

Esta apresentação é sobre metodologias ágeis?

Enfim.. BDD!

A origem


2003
2004
2006
2007
2009
agiledox, o antecessor do BDD, é uma ferramenta para geração automática de documentação técnica a partir de testes Junit, desenvolvida por Chris Stevenson
Para testar sua hipótese sobre mudar a ênfase de teste para comportamento, Dan North libera JBehave
Em colaboração com Chris Matts, North propõe *given-when-then canvas* para expandir o escopo de BDD para análise e documenta a abordagem no artigo entitulado "Introducing BDD"
Várias outras ferramentas não desenvolvidas conirmando o investimento da comunidade em BDD, ferramentas tais quais Rspec, Cucumber e GivWenZen

BDD – algumas definições

"Behaviour driven development is about implementing an application by describing its behaviour from the perspective of its stakeholders" –

Dan North, 2003

"BDD (Behaviour Driven Development) is a synthesis and refinement of practices stemming from TDD (Test Driven Development) and ATDD (Acceptance Test Driven Development)."

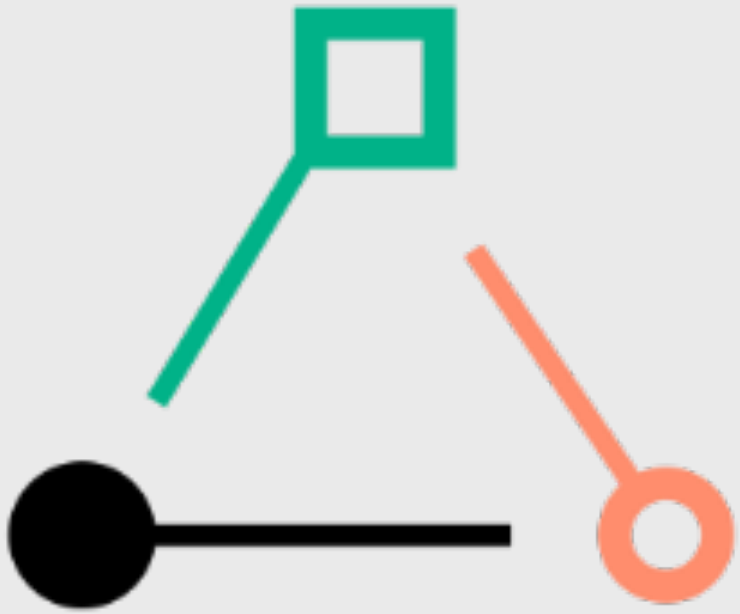
Agile alliance

"BDD is a second-generation, outside–in, pull-based, multiple-stakeholder, multiple-scale, high-automation, agile methodology. It describes a cycle of interactions with well-defined outputs, resulting in the delivery of working, tested software that matters."

Dan North, 2009

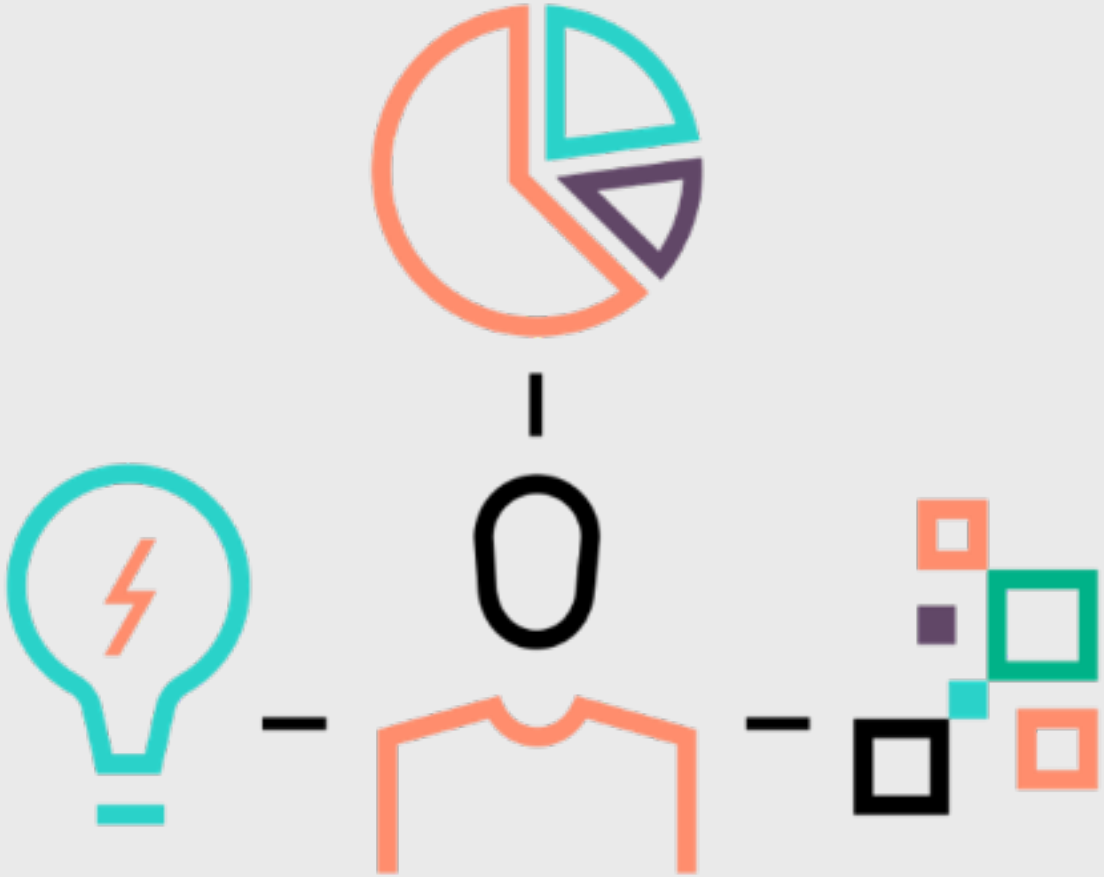
Princípios

- Apenas o **suficiente**, não mais, não menos
  - Nível necessário de detalhe (análise, design, código, estimativa, planejamento)
  - Foco em problemas existentes
- Entrega **valor** para o negócio e *stakeholders*
- O que importa é o **comportamento**
  - Em todos os níveis!
    - *Stories* e cenários descrevem comportamento no nível da aplicação
    - Exemplos de código definem comportamento no nível do código
    - …



BDD é...

um modo de pensar!



Exemplos desse modo de pensar...

Começando por um objetivo, meta, valor...

Visão

Resultado/valor

Resultado/valor

Resultado/valor

Conjunto de funcionalidades

Conjunto de funcionalidades

Conjunto de funcionalidades

Funcionalidades

Funcionalidades

Funcionalidades

…

Stories

Stories

…

Cenários

…

Entrega valor!!

Hewlett Packard Enterprise

... descritos através de *stories...*

– Uma unidade de entrega

– Entregue em uma iteração

– Tem valor

– Define comportamentos esperados

... usando exemplos ...

– Esclarece ideias
– Diminui a confusão
– Esclarece a expectativa
– Remove ambiguidade
– Ou seja cenários!

... através de uma linguagem ubíquota ...

- Conceito emprestado de DDD (Domain Driven Design)
- Línguagem comum entre técnicos e não técnicos
- Permite o entendimento comum do progresso do projeto
- Em conjunto com a descrição em termos de exemplos, endereça um dos principais desafios de desenvolvimento de software: comunicação.
  - Problemas multiplicados em caso de geografias, culturas, linguas diferentes

... com foco no usuário e considerando as diversas perspectivas!

- *Usage-centric design* garante a entrega de um software com valor para o usuário!
- A definição de uma *story* é um trabalho para “os três amigos”:
  - Representante do negócio (PO, BA, etc), desenvolvedor e testador
  - Cada um traz uma perspectiva de alto valor para esse momento
    - Valor para o negócio e avaliação de risco
    - Solução, como, quais definições precisam ser feitas, quais possibilidades
    - Problemas, falhas, “e se...”

A definição de uma story é resultado de interações, conversas, esclarecimentos, não o trabalho de uma única pessoa!

Uma *story* BDD

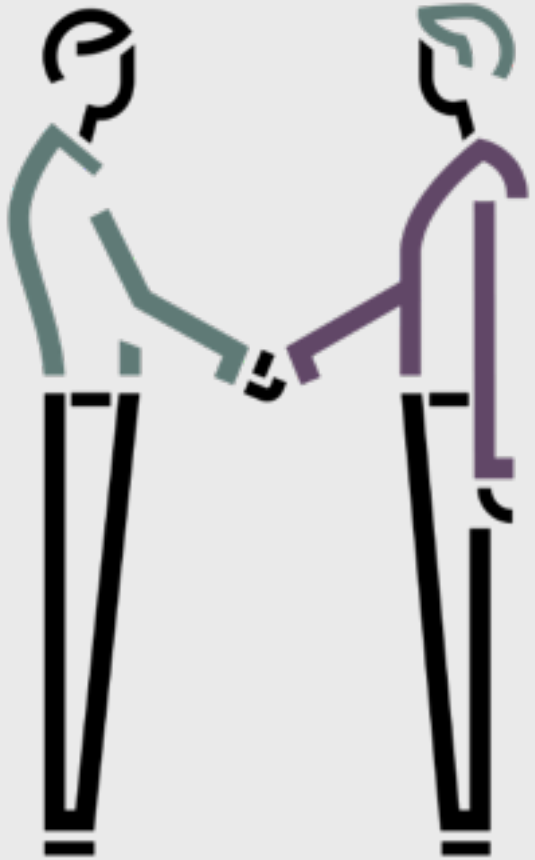
- *As a **Role***
  *I request a **Feature***
  *To gain a **Benefit***

Foco no usuário, foco no valor!

Linguagem ubíquota, usa exemplos, foco no comportamento!

- ***Given** some initial context (the givens),*
- ***When** an event occurs,*
- ***then** ensure some outcomes.*

Uma *story* é um acordo!



Stories e critérios de aceitação são o resultado e acordo gerado pela interação entre stakeholders do projeto, analistas, desenvolvedores, arquitetos, testadores, etc.

Exemplo

Funcionalidade: Itens retornados voltam ao estoque

– **Para** manter o estoque atualizado
Como dono de loja
Eu quero adicionar itens de volta ao estoque quando eles são retornados

Exemplo: Itens devolvidos devem ser retornados ao estoque

– **Dado que** um cliente comprou um casaco preto de mim
E eu tenho atualmente 3 casacos pretos no estoque
Quando ele devolve o casaco
Então eu devo ter 4 casacos pretos no estoque

Exemplo: Itens substituídos devem ser retornados ao estoque

– **Dado que** um cliente comprou uma calça azul
E eu tenho 2 calças azuis no estoque
E 3 calças pretas no estoque
Quando ele retorna a calça para trocar pela preta
Então eu devo ter 3 calças azuis no estoque
E 2 calças pertas no estoque

Linguagem *Gherkin*

- DSL
- Propósitos:
  - Documentação
  - Automação de testes
- Ferramentas:
  - Lettuce – Python
  - Bhave - Python
  - jBehave - Java
  - Cucumber – Ruby, Java, Javascript, C++
  - Behat - PHP
  - SpecFlow - .NET, Mono, Silverlight, WP7
  - ...

Resumindo

- Muda o foco dos “testes” (*tests*) para “comportamento” (*behavior*)
- Reforça a colaboração entre stakeholders, analistas, desenvolvedores, testadores, product owner, gerente de projetos, …
- Usa uma línguagem ubíquota e que os envolvidos entendem
  - Nada de tecniquês!
- Foca no valor para o negócio!
- Extende TDD através do uso de linguagem natural que pode ser entendida por pessoas não-técnicas envolvidas no projeto

Voltando a definição inicial

Definição Dan North, 2009	Re-leitura
– second-generation, – outside–in, – pull-based, – multiple-stakeholder, – multiple-scale, – high-automation, – agile methodology.	– re-leitura de TDD, DDD, Lean... – começa pela visão, valor do negócio – só o suficiente, sem excesso – centrado no usuário e envolve stakeholders (todos que se importam) – diversos níveis (aplicação, código) – automação, regressão, TDD – princípios e valores compartilhados

Uma implementação

- Sintaxe Gherkin:
  - Lettuce (Python), melhor suporte a DJango
- Automação de testes interface web (corpo do método usa python):
  - Selenium
- Gerrit (code review) + Jenkins
  - A cada commit os testes BDD são executados

Destaque

Vale lembrar que...

- Não é bala de prata!
- Testes manuais são sempre necessários (exploratórios, usabilidade, novos cenários)
- BDD não é apenas sobre UI!!
- BDD não é sobre o uso de ferramentas!
- BDD é modo de pensar... E isso inclui gerência!
- Foco na colaboração
- Facilita comunicação através de entendimento comum!

E eu com isso?

Hewlett Packard
Enterprise

Obrigada!

glaucimar.aguiar@hpe.com