

# BDD além do Hello World

#TDC2011 Goiania  
Eder Ignatowicz

# Audiência #TDC2011

## Trilha de Testes



PO / Cliente

# Audiência #TDC2011

## Trilha de Testes



PO / Cliente



Tester

# Audiência #TDC2011

## Trilha de Testes



PO / Cliente



Tester



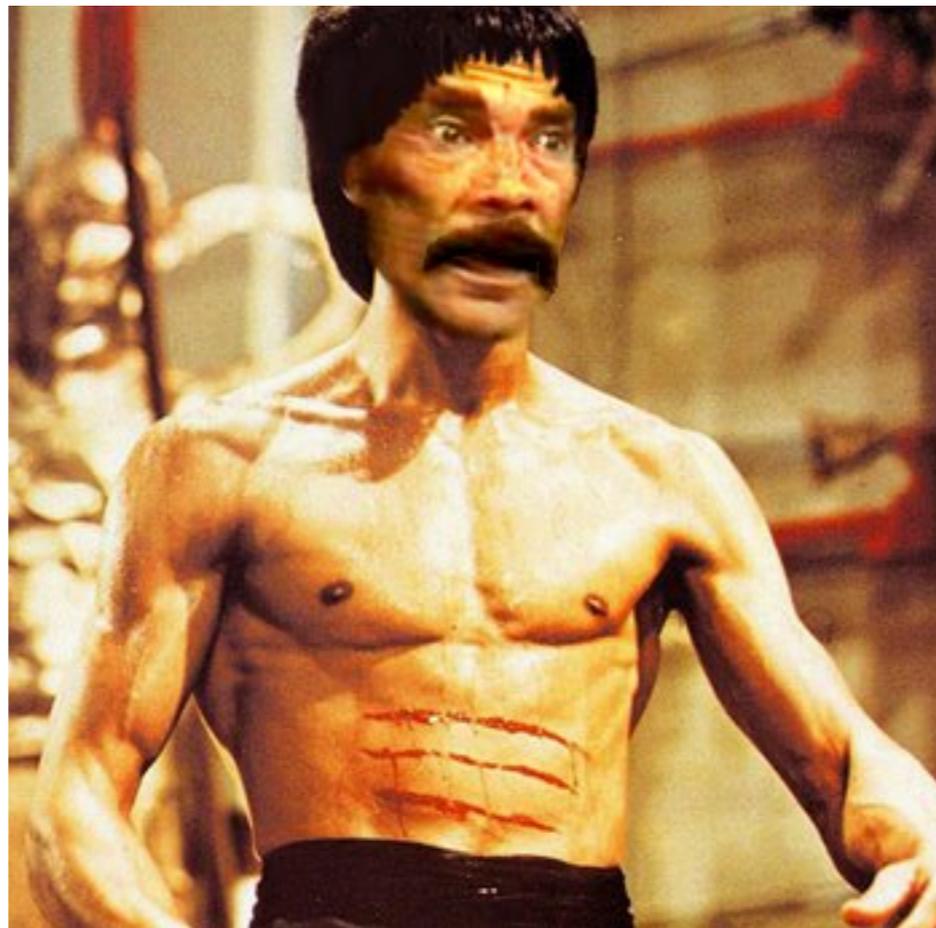
Developer

# Eder Ignatowicz



Developer?

# Eder Ignatowicz



**Multidisciplinar**

# Eder Ignatowicz





- Unidades:**
- São Paulo
  - Campinas
  - Campo Grande



No princípio...

```

while ((!found) && (pos < (fileContent.Length - 6)))
{
    byteData = new byte[6];
    Array.Copy(fileContent, pos, byteData, 0, 6);
    pos = pos + 6;
    str_byteData = enc.GetString(byteData);
    if (str_byteData.Contains("s"))
    {
        posE_byteData = str_byteData.IndexOf("s");
        pos = pos + (posE_byteData - 6);
        Array.Copy(fileContent, pos, byteData, 0, 6);
        pos = pos + 6;
        if (byteData[0] == 0x73) // 's'
        {
            if (byteData[1] == 0x74) // 't'
            {
                if (byteData[2] == 0x72) // 'r'
                {
                    if (byteData[3] == 0x65) // 'e'
                    {
                        if (byteData[4] == 0x61) // 'a'
                        {
                            if (byteData[5] == 0x6D) // 'm'
                            {
                                found = true;
                                break;
                            }
                        }
                    }
                }
            }
        }
        else
        {
            if (byteData[5] == 0x73)
            {
                pos = pos - 1;
            }
        }
    }
}

```

**Como “testar” isto?**





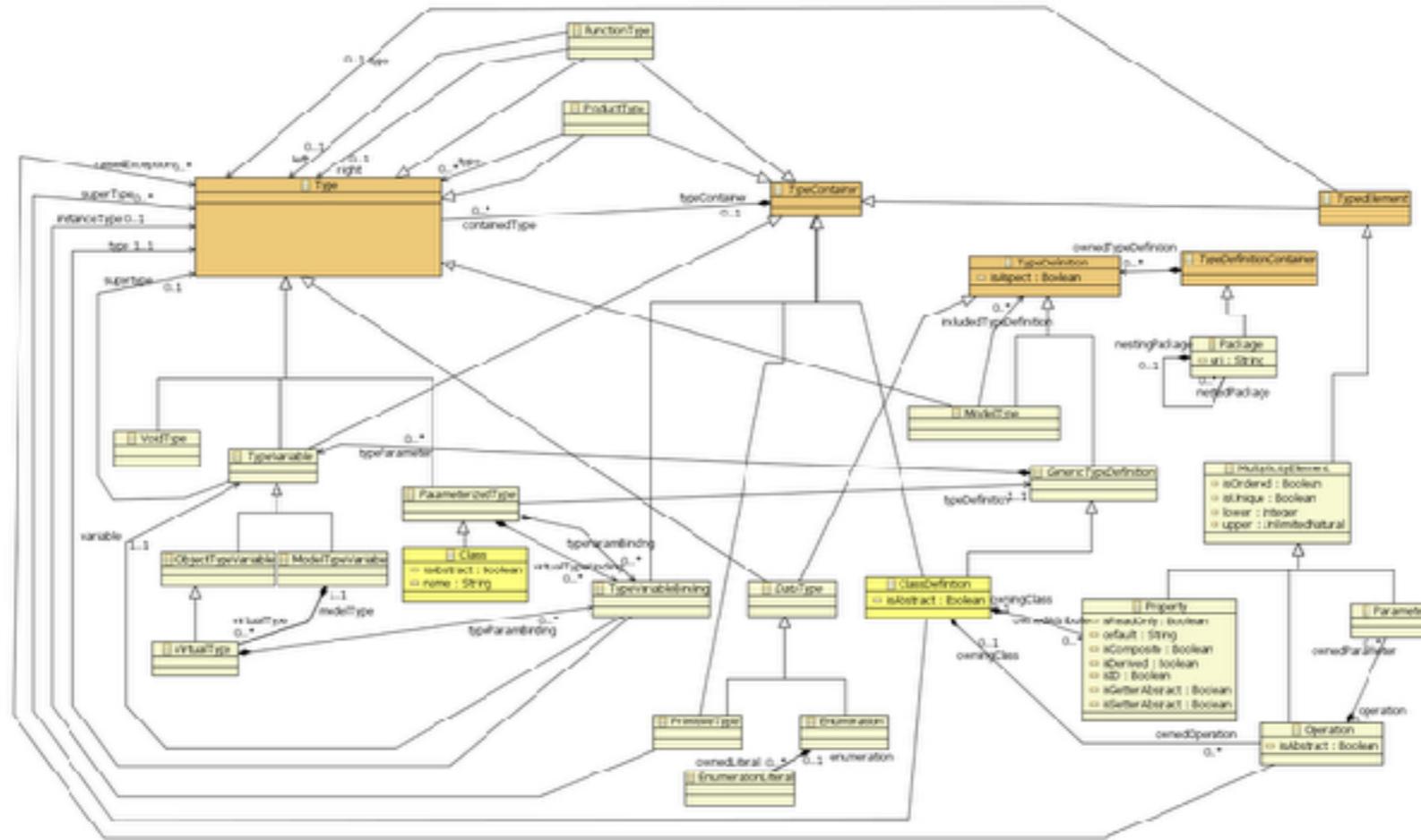
Kent Beck “enviou” o TDD aos programadores

**É o fim dos nossos  
problemas?**



Eu sou um programador, não  
um tester...  
“Na minha máquina  
funciona”





Meu código é difícil de  
testar...



**É trivial, não preciso  
testar**

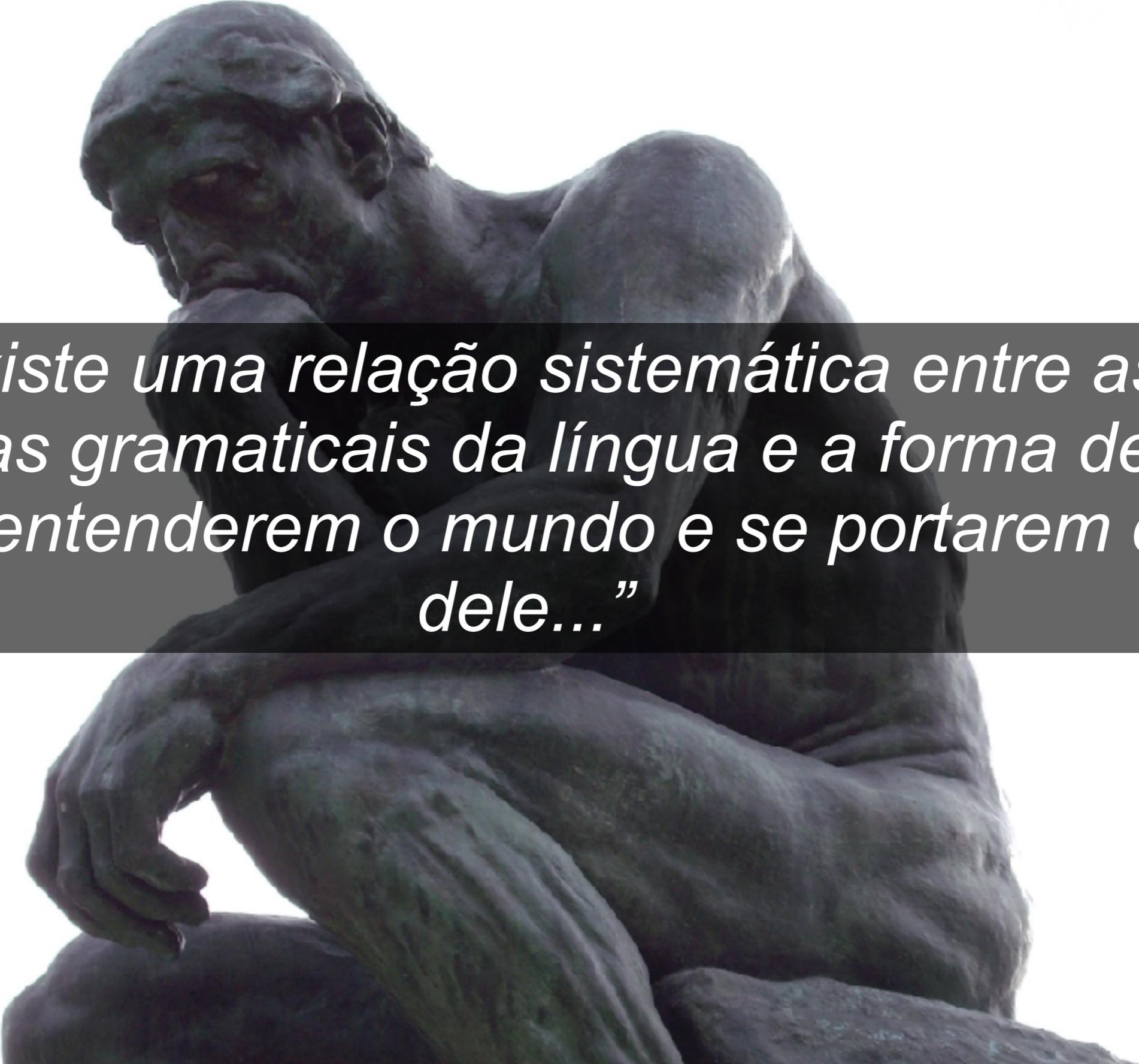


Preciso de uma semana para  
executar os testes de  
regressão



**Eu não tenho tempo para  
automatizar os testes**

# Hipótese Sapir-Whorf



*“existe uma relação sistemática entre as categorias gramaticais da língua e a forma de seus falantes entenderem o mundo e se portarem dentro dele...”*

**It's all behaviour!!!**



Dan North

# BDD: Behaviour Driven Development

“Escrevendo **SOFTWARE** e não código”

# Princípios do BDD

*It's all behaviour*



**Com objetivo de** acelerar a cobrança do aluguel  
**Como** um cobrador  
**Eu quero** visualizar o total da dívida de um locatário

Descrição da história (Wiki, JIRA, Tracker, Word)

# Princípios do BDD

*It's all behaviour*



```
Assert.assertEquals(locatario.di  
vidaTotal(), 17000,00);
```

Testes automatizados de unidade e aceitação

# Princípios do BDD

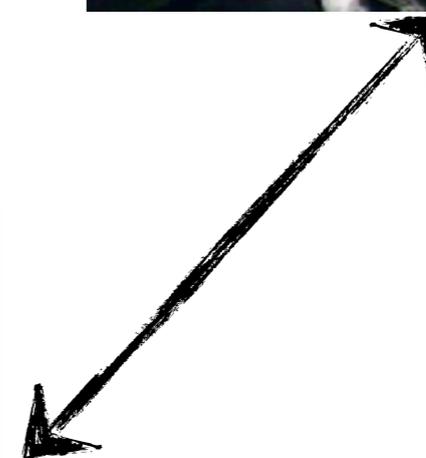
*It's all behaviour*



- 1-) Devo um aluguel de R\$ 1200,00**
- 2-) Devo um aluguel de R\$ 1200,00**
- 3-) O total da dívida deve ser R\$ 2400,00**

Procedimento de Testes

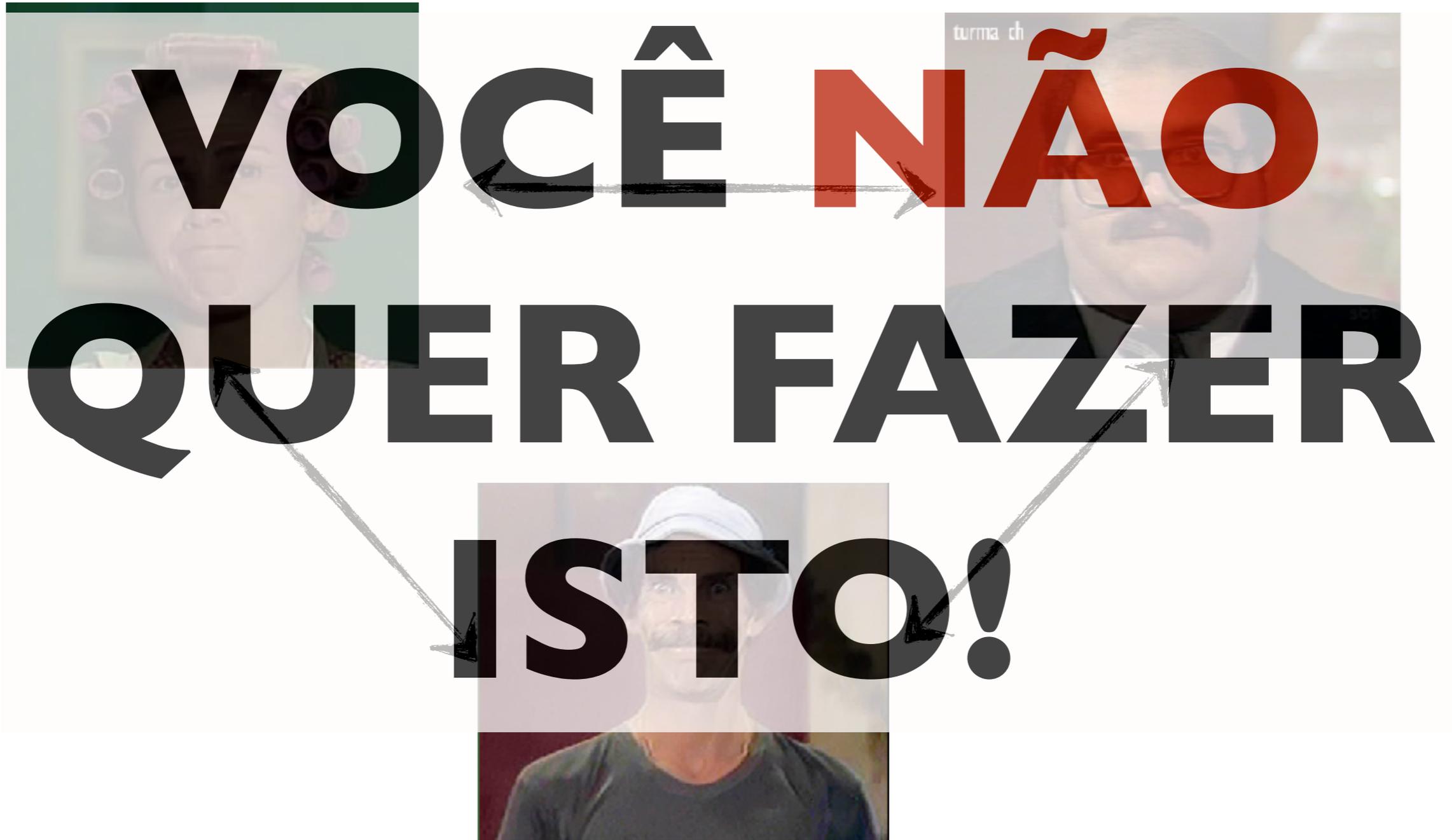
# Tradução



# Tradução

**VOCÊ NÃO QUER FAZER**

**ISTO!**



# Linguagem Comum

**Title** (one line describing the story)

**Narrative:**

As a [role]

I want [feature]

So that [benefit]

Acceptance Criteria: (presented as Scenarios)

**Scenario 1:** Title

Given [context]

And [some more context]...

When [event]

Then [outcome]

And [another outcome]...

Scenario 2: ...

# Linguagem Comum

**Dado (Given)** uma conta de aluguel

**Quando (When)** o locatário deve um aluguel  
no valor de R\$1200,00

**E (And)** o locatário deve outro aluguel no valor  
de R\$1200,00

**Então (Then)** o total da dívida é R\$2400,00



# Falando a mesma linguagem....



Nós podemos nos **entender** melhor;

Nós podemos nos **aprofundar** no  
**entendimento** do problema;

É maximizar as chances de **construirmos**  
as coisas **certas**.

# Princípios do BDD

## *Deliver stakeholder value*



**Com objetivo de** acelerar a cobrança do aluguel, **por quê?**

**Com objetivo de** controlar melhor a cobrança de juros, **por quê?**

**Com objetivo de** controlar melhor os meus resultados financeiros, **BAZINGA!**

# Redução de Custos



Eu perco menos **tempo** e **dinheiro** construindo coisas que não são exatamente necessárias e investigando valores que me ajudam a **priorizar corretamente!**

# Princípios do BDD

## *Enough is Enough*



Ficou fácil descobrir onde se inicia e termina uma funcionalidade.



Parei de desperdiçar dinheiro com funcionalidades supérfluas

# Detalhamento correto

**Dado (Given)** uma conta de aluguel

**Quando (When)** o locatario deve um aluguel  
no valor de R\$1200,00

**E (And)** o locatário deve outro aluguel no valor  
de R\$1200,00

**Então (Then)** o total da dívida é R\$2400,00

# Detalhamento correto...



Eu tenho algo para construir o mais rápido possível!



Eu tenho algo para testar rapidamente!

# Diminuindo a Carga no Final do Sprint



# Explore no início



Chance menor de ser surpreendido e tempo de descobrir mais problemas



Bugs e novos cenários rapidamente

# SubProdutos do BDD



**Rede de Segurança**

# Testes Automatizados

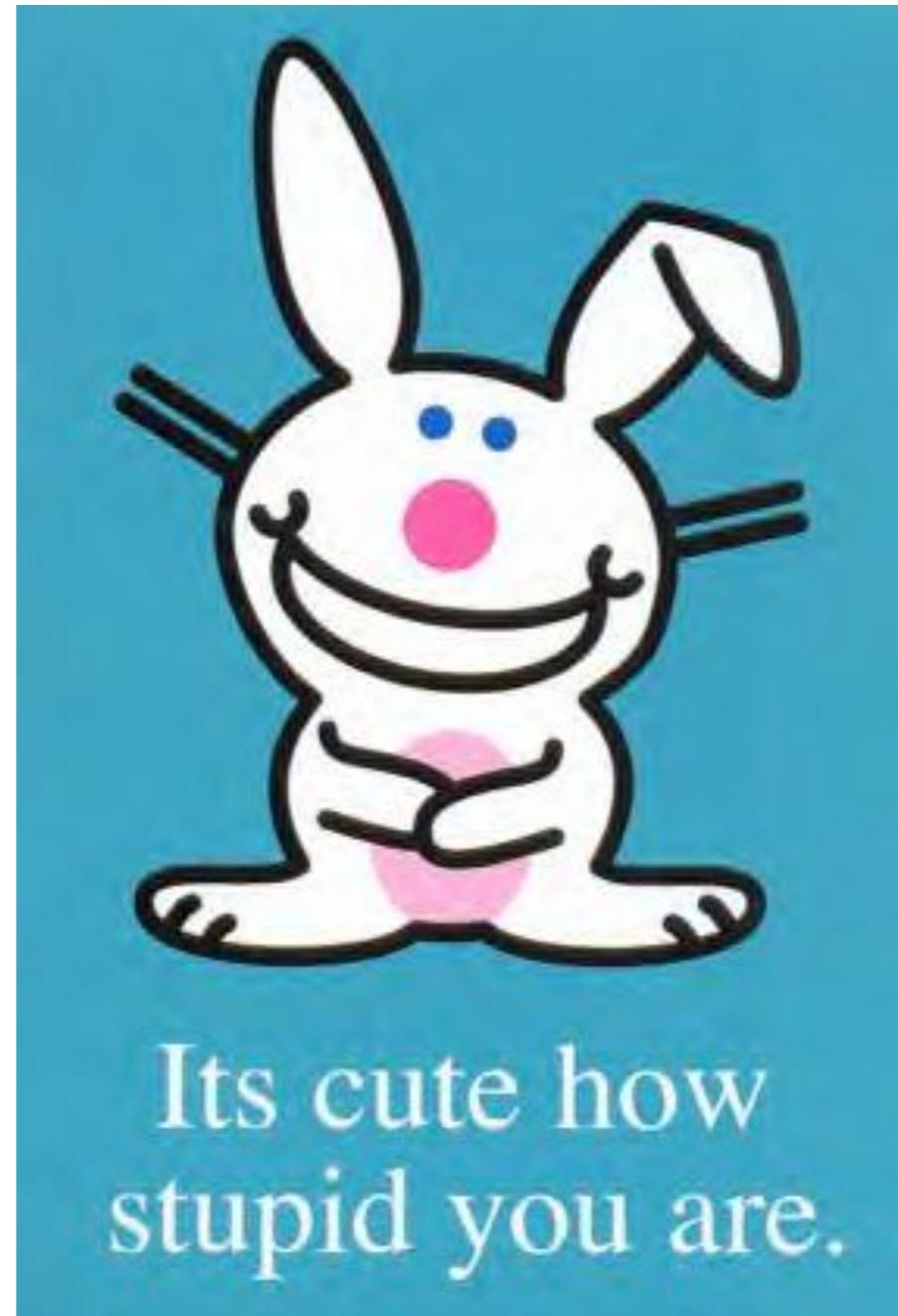


Eu posso fazer modificações com confiança e velocidade



Hummm... eu posso utilizar estes testes?

# Testes de Regressão Manual



# Propriedade Coletiva



Features + Definição de Passos



# Suíte de Regressão Automática



Testes de regressão ficaram baratos!  
Posso gerar versões mais rapidamente



Eu posso me focar em mais testes  
exploratórios!

# Especificação “Tradicional”



# Especificação executável

**Dado (Given)** uma conta de aluguel

**Quando (When)** o locatário deve um aluguel  
no valor de R\$1200,00

**E (And)** o locatário deve outro aluguel no valor  
de R\$1200,00

**Então (Then)** o total da dívida é R\$2400,00

**Resumindo, porque BDD?**

Melhor entendimento do problema  
Chance de melhorar meu investimento  
Desenvolvimento mais eficiente  
Feedback rápido  
Documentação "viva"



Melhor entendimento do problema  
Eu sei onde inicia, o que fazer em  
seguida e quando parar o  
desenvolvimento de uma feature  
Chance de construir algo rapidamente  
Rede de segurança para efetuar  
mudanças



Melhor entendimento do problema  
Software para testar mais  
rapidamente  
Mais tempo para testes  
exploratórios



**JBehave**

# JBehave

"Behaviour-driven development in Java..."

## Philosophy

### JBehave philosophy

The philosophy that drove the development of JBehave is the same that underlies **BDD**, i.e. to facilitate the **communication** of business requirements or behaviours between the stakeholders and the development team, and to allow these behaviours to be verified in an automated way using **continuous integration**.

The emphasis here is on *communication*.

# JBehave

Title (one line describing the story)

Narrative:

As a [role]

I want [feature]

So that [benefit]

**Advantages of the “As a user, I want” user story template, by Mike Cohn**

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title

**Given** [context]

**And** [some more context]...

**When** [event]

**Then** [outcome]

**And** [another outcome]...

**Cenários descrevendo os comportamentos esperados.**

Scenario 2: ...

# JBehave

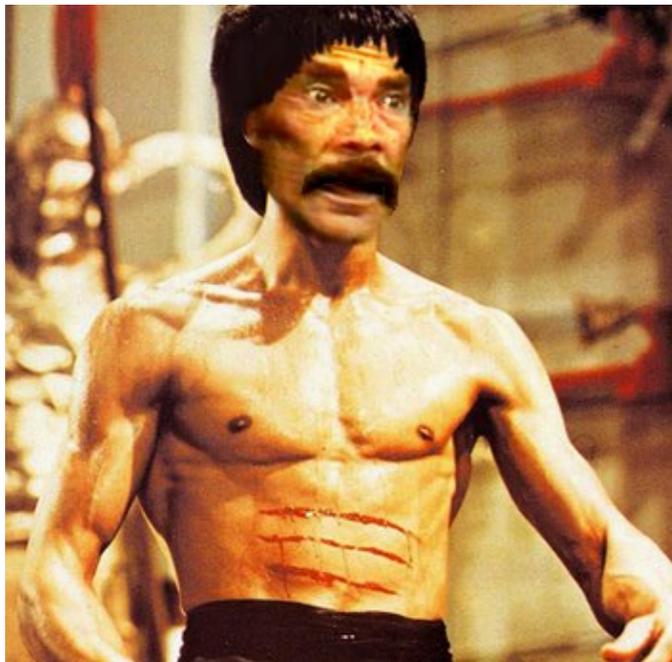
Utilização das palavras chave dos cenários  
em texto puro;

Criação classes para mapear os passos  
(Steps);

Criação de classes para rodar os cenários,  
chamando os steps necessários;

# Live Coding - JBehave

JBehave != BDD



“All of these tools are great... but, in the end, tools are tools. While RSpec and Cucumber are optimized for BDD, using them doesn't automatically mean you're doing BDD"

The RSpec Book

# Dúvidas?



Eder Ignatowicz  
ignatowicz@gmail.com



@ederign

## Bibliografia

Apresentação baseado no trabalho de Brent Snook