

Requisitos Executáveis: O modelo FIT / FitNesse

Jorge Diz
Globalcode

Jorge Alberto Diz

- Mestre em Eng. Elétrica (UNICAMP ´95)
- Bach. em Ciência da Computação (UNICAMP ´89)
- Programando desde ´83, em Java desde ´99
- Automação de testes desde ´94
- Ensinando desde ´01, na Globalcode desde ´06
- Certificado SCJP, SCWCD, CSM
- Consultor em metodologias ágeis

Especificação X requisito X teste X exemplo

- A pirâmide alimentar: o recheio do sanduiche
- Genealogia do FitNesse e ATDD
- Arquitetura
- Fixtures prontas
- Fixtures programadas
- TDD, ATDD, BDD
- Conclusão e caminhos futuros

Qual a diferença entre ... ?

- **Especificação funcional**
- **Requisito**
- **Requisito de teste**
- **Exemplo**
- **Cenário de teste**
- **Caso de teste**

Qual a diferença entre ... ?



requisitos



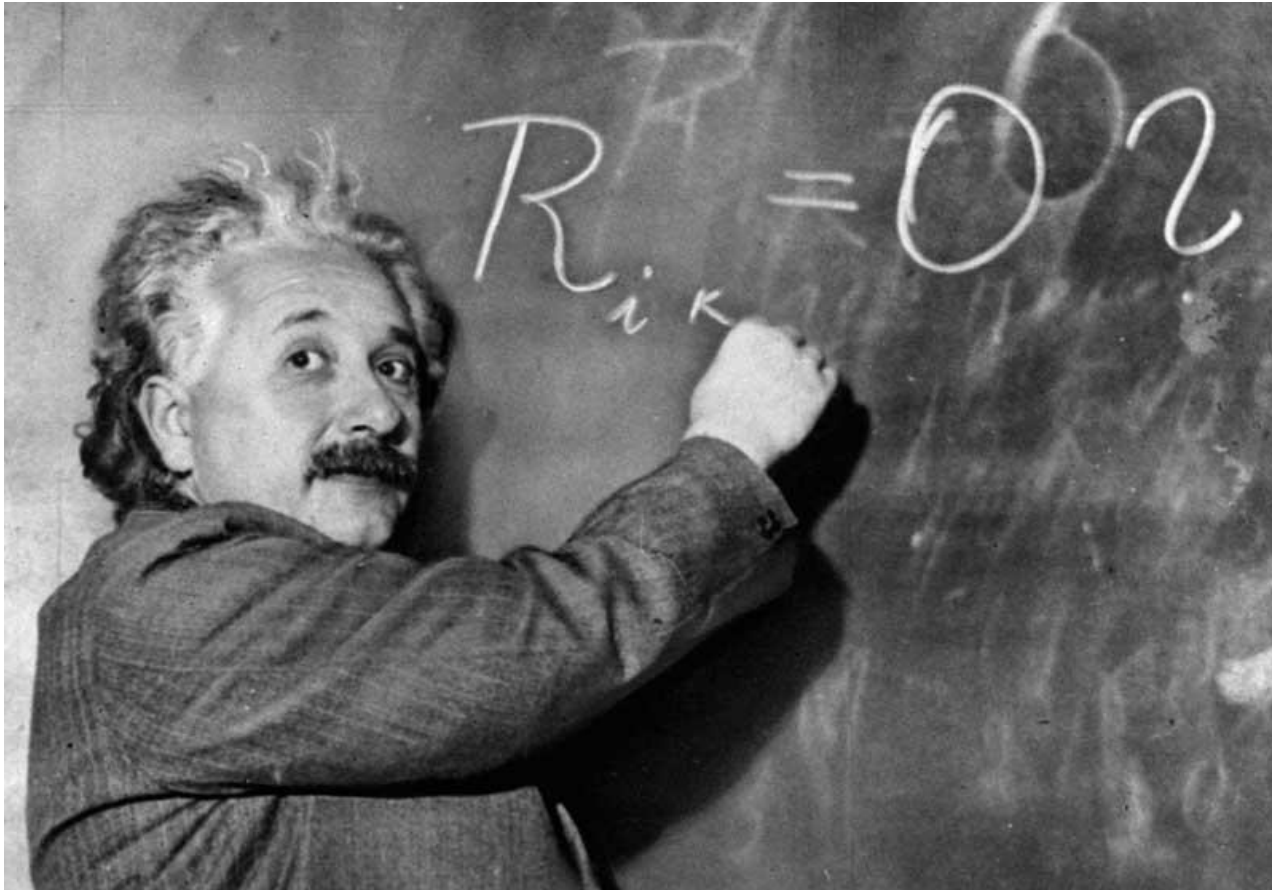
testes



exemplos

⇒ **Convergência de artefatos**

- **menos ruído na comunicação**
- **favorece uma linguagem comum**



”Exemplos não são uma outra forma de ensinar:
são a única forma”

Developer Tests:

Olhando para a tecnologia

Testes técnicos, de componentes

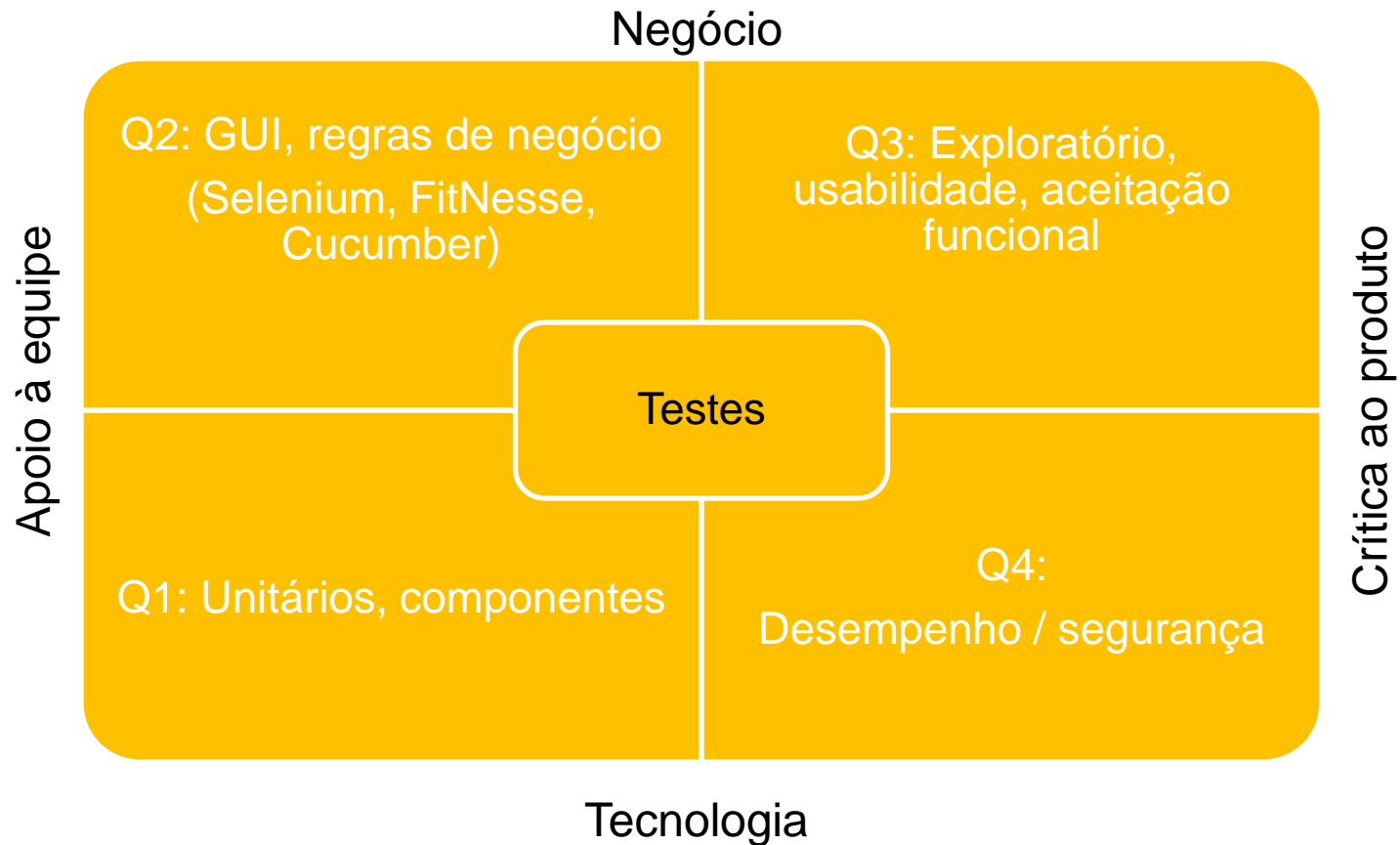
Ferramentas: XUnit, mocks, ...

Customer Tests:

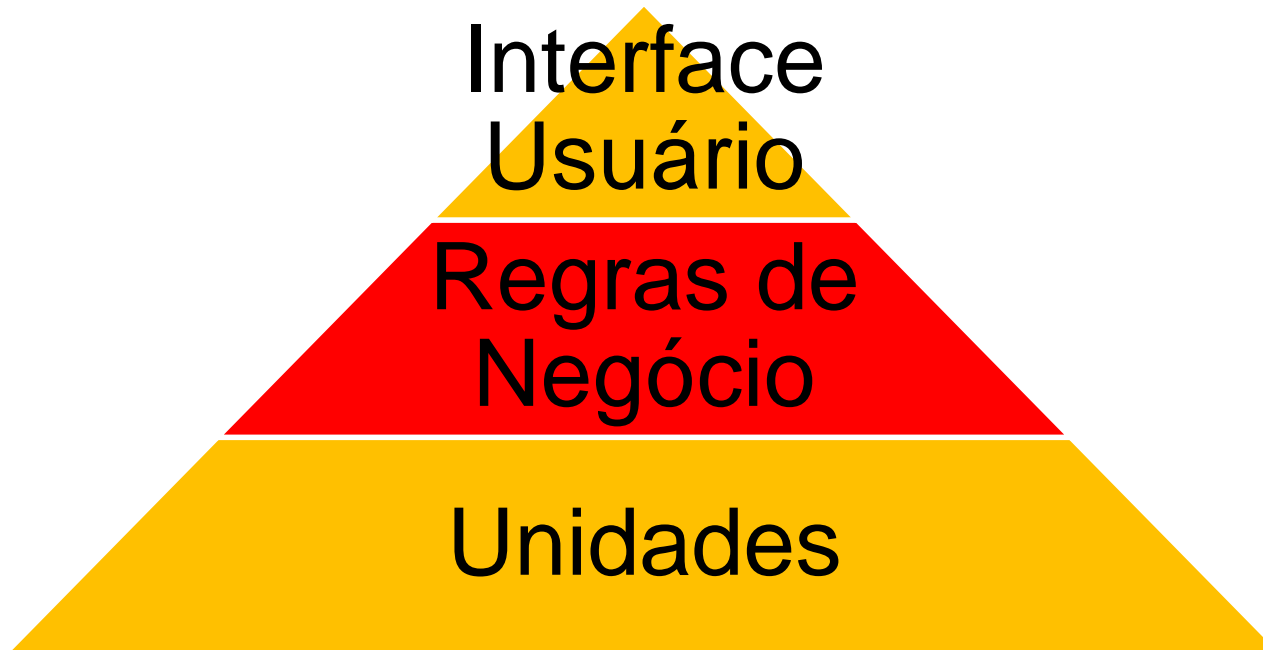
Olhando para o negócio

Testes de aceitação, regras de negócio, GUI

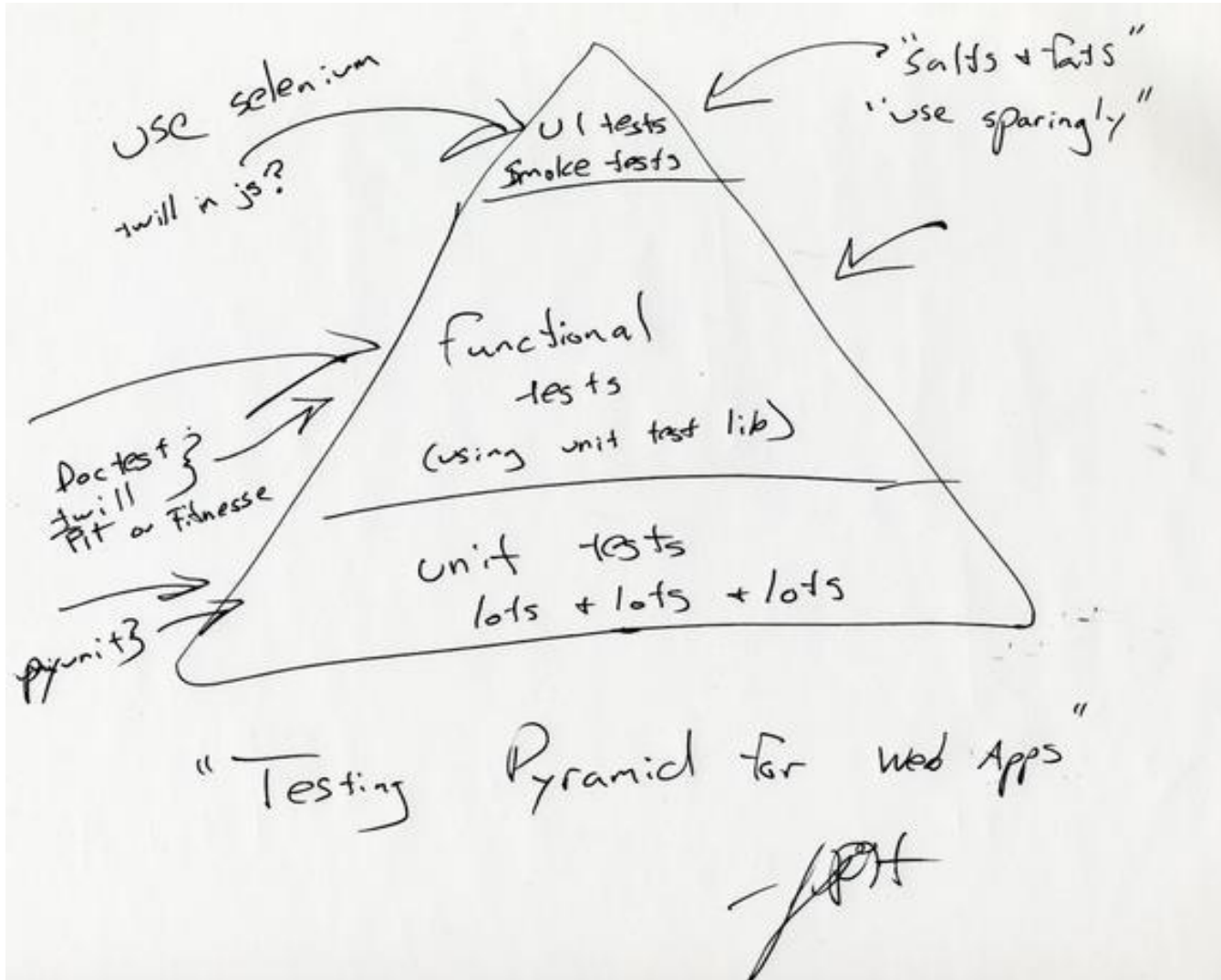
Ferramentas: FIT / FitNesse, Cucumber, Selenium







Pirâmide de testes by Huggins *

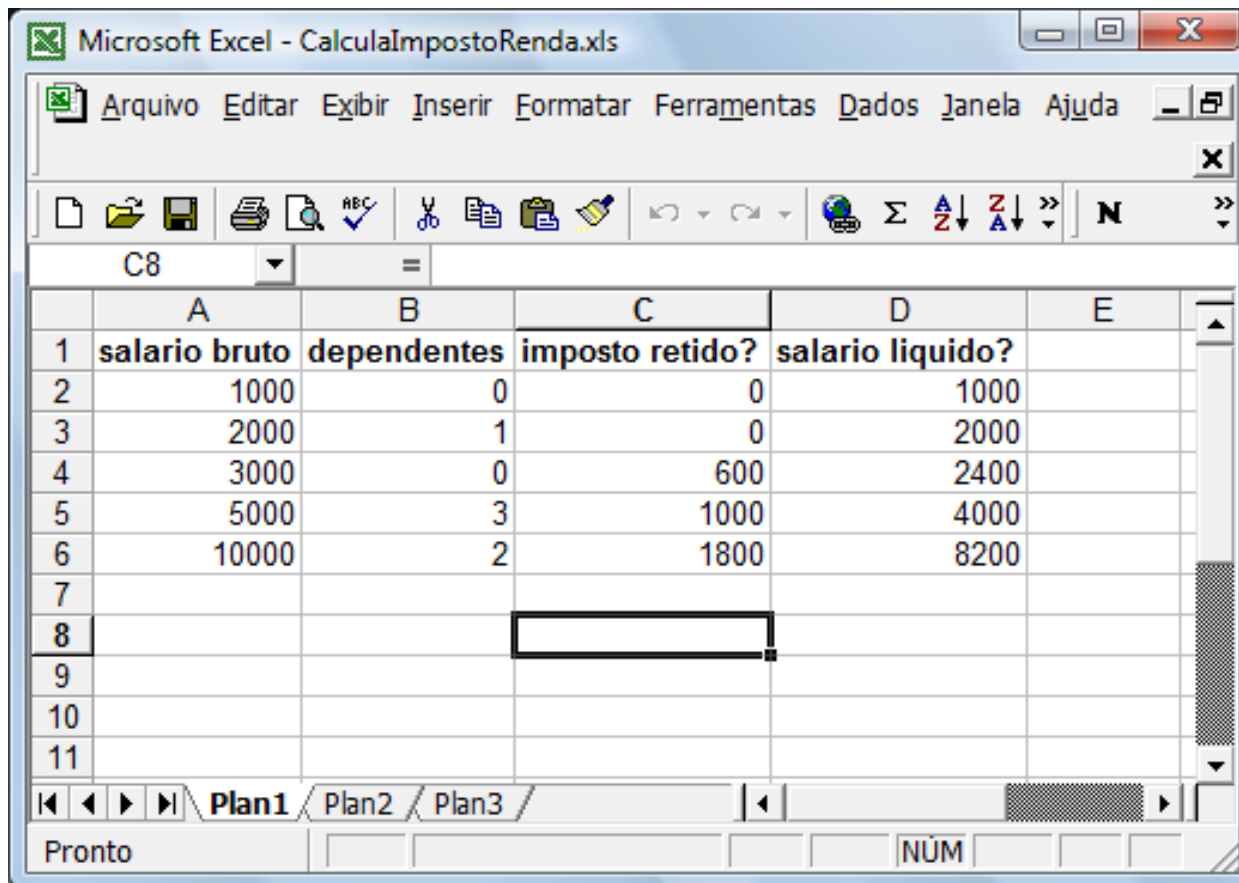


(*) Jason Huggins, autor do Selenium

A pirâmide alimentar



- **Usuário descreve exemplos de regras de negócio em planilhas (Excel, OpenOffice)**
- **Natural para vários perfis de usuários**
- **Planilhas são traduzidas para tabelas HTML**
- **Tabelas HTML são interpretadas por classes Java**



Microsoft Excel - CalculaImpostoRenda.xls

Arquivo Editar Exibir Inserir Formatar Ferramentas Dados Janela Ajuda

C8 =

| | A | B | C | D | E |
|----|---------------|-------------|-----------------|------------------|---|
| 1 | salario bruto | dependentes | imposto retido? | salario liquido? | |
| 2 | 1000 | 0 | 0 | 1000 | |
| 3 | 2000 | 1 | 0 | 2000 | |
| 4 | 3000 | 0 | 600 | 2400 | |
| 5 | 5000 | 3 | 1000 | 4000 | |
| 6 | 10000 | 2 | 1800 | 8200 | |
| 7 | | | | | |
| 8 | | | | | |
| 9 | | | | | |
| 10 | | | | | |
| 11 | | | | | |

Plan1 Plan2 Plan3

Pronto NUM

- **FIT (Framework for Integration Testing)**
- **Wiki**
- **FIT + Wiki = FitNesse**
- **FitNesse 2009mmdd = (FIT | SLIM) + Wiki**

- **TDD (test driven development) = test-first + refactoring**
- **ATDD = Acceptance TDD =**
 - **Testes de aceitação como guia para desenvolver**
 - **“Requisitos executáveis”**
- **BDD = Behavior Driven Development)**
 - **ATDD + business personas + formalismos**

Complicadores:

- Disponibilizar (ou aproximar) o ambiente de servidor web
- Dependência de configurações
- Dependência de browsers
- Dependência de JavaScript
- AJAX / tempos de espera

Moral da história:

- Muita dor de cabeça no teste de UI
- Relação custo/benefício do teste de UI é menos favorável
- Precisamos focar na lógica de negócio

- **O quê estamos testando ?**
 - Módulos, classes isoladamente ---> X(J)Unit
 - **Regras de negócio → FIT / FitNesse**
 - Funcionalidade de componentes web → Cactus (in container)
 - Interface usuário--> Selenium (in browser)
 - Estresse/desempenho ---> JMeter

FitNesse - arquitetura

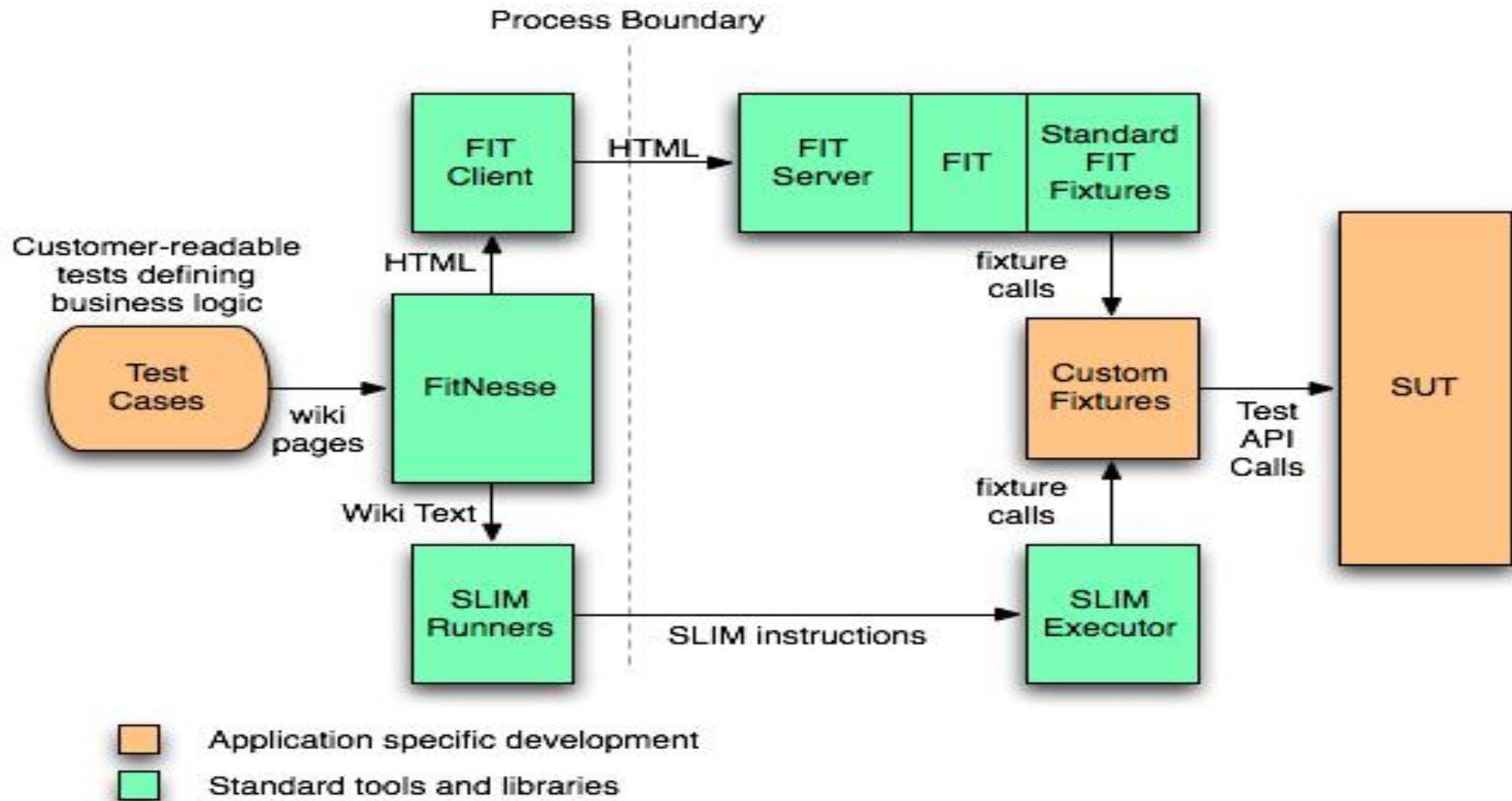
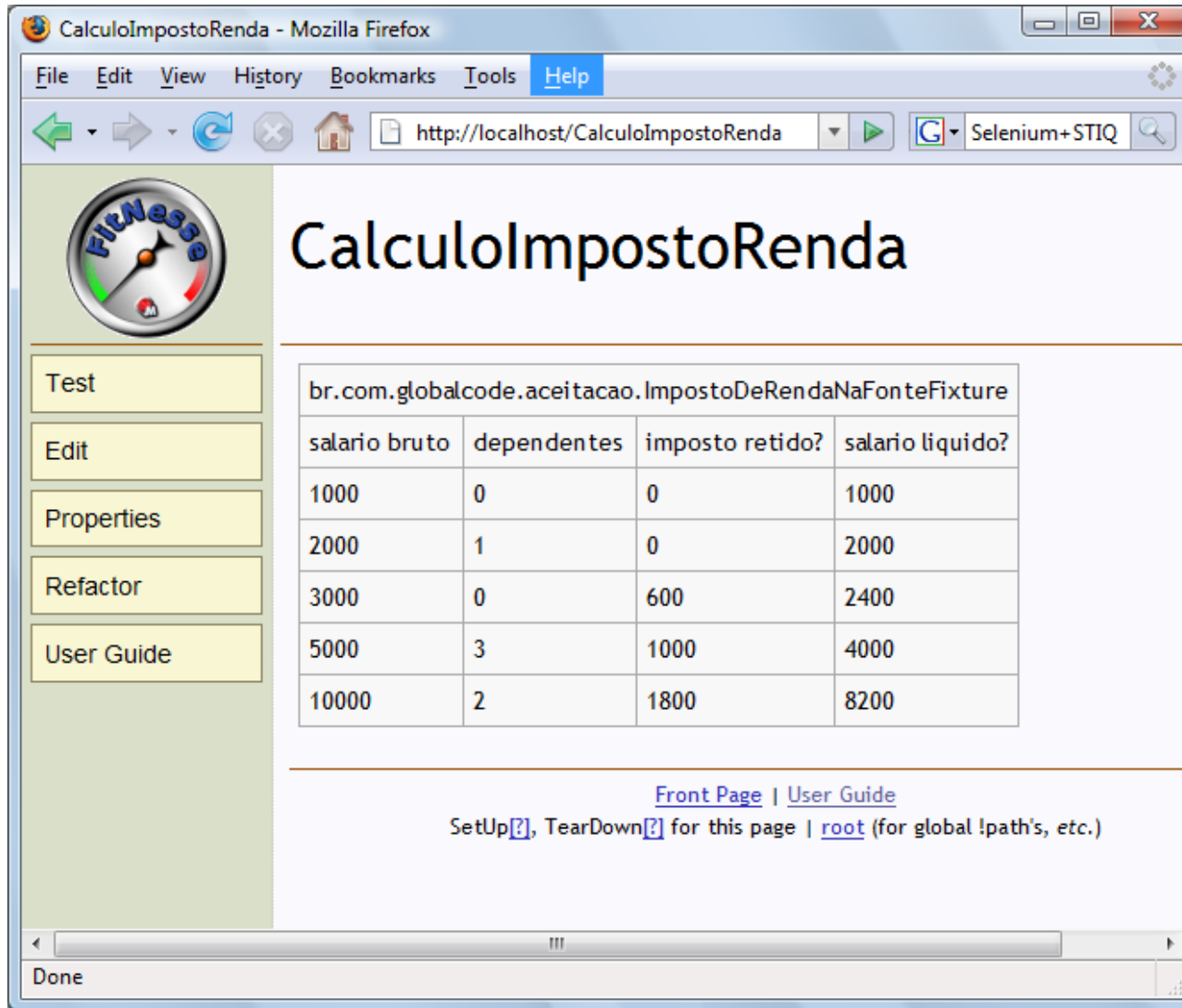


diagrama extraído do site <http://fitnesse.org>

- Classes interpretadoras de tabelas
- Cada uma implementa uma interpretação diferente.
- Escritas em uma linguagem de programação (Java, C#, Python, Ruby, ...)
- Acionam a lógica das classes do sistema sendo testado.

FitNesse – tabela Wiki



The screenshot shows a Mozilla Firefox browser window with the address bar at `http://localhost/CalculoImpostoRenda`. The page title is "CalculoImpostoRenda". On the left side, there is a sidebar with a "FitNesse" logo and several buttons: "Test", "Edit", "Properties", "Refactor", and "User Guide". The main content area displays a table with the following data:

| salario bruto | dependentes | imposto retido? | salario liquido? |
|---------------|-------------|-----------------|------------------|
| 1000 | 0 | 0 | 1000 |
| 2000 | 1 | 0 | 2000 |
| 3000 | 0 | 600 | 2400 |
| 5000 | 3 | 1000 | 4000 |
| 10000 | 2 | 1800 | 8200 |

Below the table, there are links for "Front Page" and "User Guide", and a note: "SetUp, TearDown for this page | root (for global !path's, etc.)". The status bar at the bottom of the browser window shows "Done".

```
package br.com.globalcode.aceitacao;

import fit.ColumnFixture;
import br.com.globalcode.impostos.RendaNaFonte;

public class ImpostoDeRendaNaFonteFixture extends ColumnFixture{

    public double salarioBruto;
    public int dependentes;

    public double impostoRetido() {
        return RendaNaFonte.desconto(salarioBruto);
    }

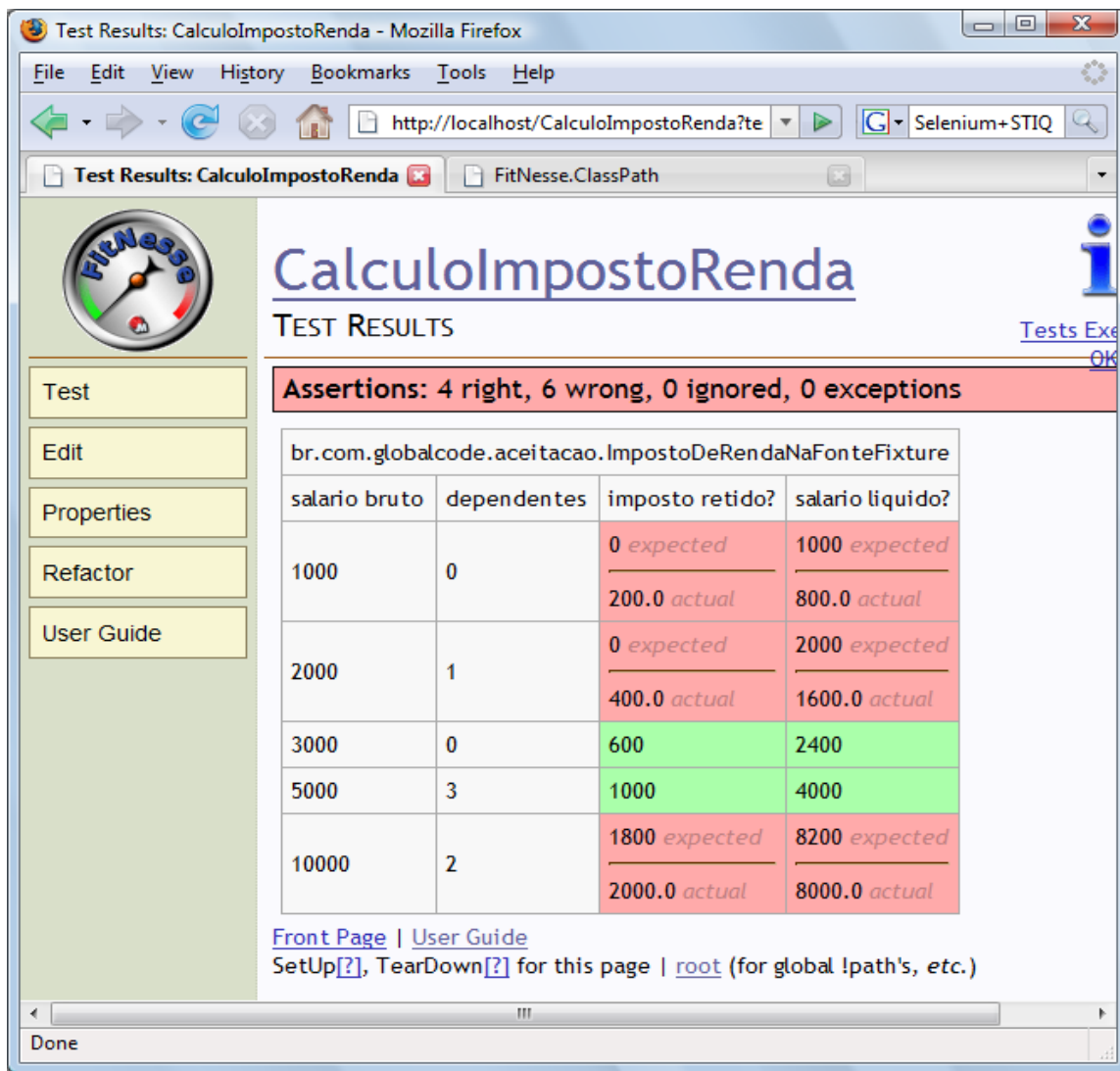
    public double salarioLiquido() {
        return RendaNaFonte.liquido(salarioBruto);
    }
}
```

```
package br.com.globalcode.impostos;

public class RendaNaFonte {
    public static double desconto(double bruto) {
        return bruto * 0.2;
    }

    public static double liquido(double bruto) {
        return bruto * 0.8;
    }
}
```

FitNesse – resultado




Test Results: CalculoImpostoRenda - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost/CalculoImpostoRenda?te Selenium+STIQ

Test Results: CalculoImpostoRenda FitNesse.ClassPath



CalculoImpostoRenda

TEST RESULTS

Assertions: 4 right, 6 wrong, 0 ignored, 0 exceptions

| salario bruto | dependentes | imposto retido? | salario liquido? |
|---------------|-------------|--|--|
| 1000 | 0 | 0 <i>expected</i> 200.0 <i>actual</i> | 1000 <i>expected</i> 800.0 <i>actual</i> |
| 2000 | 1 | 0 <i>expected</i> 400.0 <i>actual</i> | 2000 <i>expected</i> 1600.0 <i>actual</i> |
| 3000 | 0 | 600 | 2400 |
| 5000 | 3 | 1000 | 4000 |
| 10000 | 2 | 1800 <i>expected</i> 2000.0 <i>actual</i> | 8200 <i>expected</i> 8000.0 <i>actual</i> |

[Front Page](#) | [User Guide](#)
Setup[?], TearDown[?] for this page | [root](#) (for global !path's, etc.)

Done

Demo

Não precisam de programação

- DBFit
 - Banco de dados
- Generic Fixture
 - Acesso direto às classes de negócio (reflexão)
- PlainSeleniumFixture, WebTest
 - Scripts para teste de interface usuário

- É necessário escrever numa linguagem de programação (Java e outras)
- ColumnFixture
 - Planilha com colunas de entrada e resultados
 - RowFixture
 - Similar à anterior, com repetição em linhas
 - ActionFixture, DoFixture
 - Criação de linguagens específicas de domínio
 - TableFixture
 - Fixture que permite interpretações customizadas

- DSL = “domain-specific language”
- Linguagens específicas para um determinado domínio de aplicação. Ex: teste de GUI, seguro de automóvel
- Criadas caso-a-caso, aproveitam o motor do FIT / Slim
- Podem ser implementadas utilizando fixtures customizadas (DoFixture)
- Podem suportar BDD

- Promove colaboração de analistas de negócio, testers, desenvolvedores
- Front-end para múltiplos usos
- Motor de testes sem front-end (Trinidad) apto para integração contínua
- Versionável (casos de teste em formato texto)

Email: jorge@globalcode.com.br

Email, Gtalk: jorge.a.diz@gmail.com

- Twitter: [@jorgediz](https://twitter.com/jorgediz)
- www.globalcode.com.br