TESTES DE SOFTWARE & TDD

ENGENHARIA DE SISTEMAS DE INFORMAÇÃO

Daniel Cordeiro

6 de outubro de 2017

Escola de Artes, Ciências e Humanidades | EACH | USP

DEPURAR VS. TESTAR

Everyone knows that debugging is twice as hard as writing a program in the first place. So if you're as clever as you can be when you write it, how will you ever debug it?

Brian Kernighan, *The Elements of Programming Style*

Program testing can be used to show the presence of bugs, but never to show their absence!

Edsger W. Dijkstra, *Notes On Structured Programming*

Antes

- desenvolvedores terminavam de escrever o código e faziam alguns testes *ad hoc*
- código era “jogado” pro pessoal de Quality Assurance (QA)
- pessoal de QA mexia manualmente no programa

TESTE MODERNO

Antes

- desenvolvedores terminavam de escrever o código e faziam alguns testes *ad hoc*
- código era “jogado” pro pessoal de Quality Assurance (QA)
- pessoal de QA mexia manualmente no programa

Hoje/Ágil

- teste é parte de todas as iterações Ágil
- desenvolvedores testam o seu próprio código
- ferramentas & processos de testes totalmente automatizados
- equipe de testes/QA encarregada de melhorar as ferramentas e testabilidade do código

A qualidade do software é resultado de um bom processo e não a responsabilidade de um grupo específico.

Projeto guiado por comportamento (BDD)

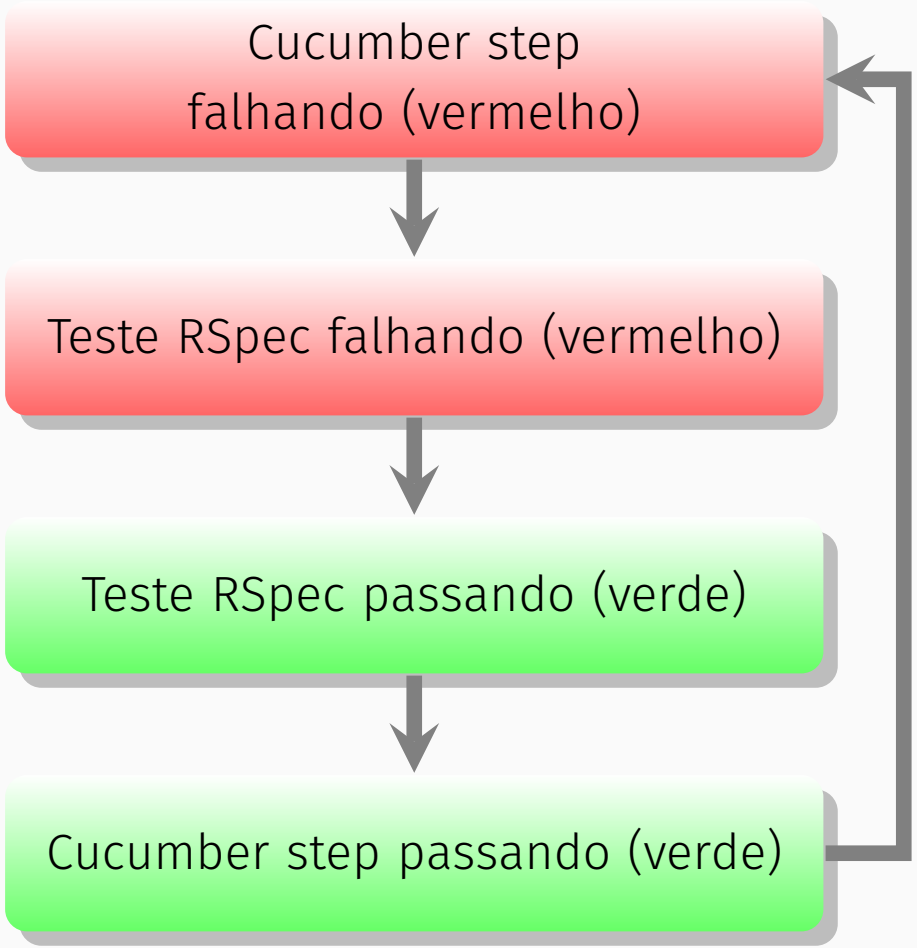
- desenvolva histórias de usuários (as funcionalidades que você quer ter) para descrever como o app irá funcionar
- usando Cucumber, histórias de usuários viram *testes de aceitação* e *testes de integração*

Desenvolvimento guiado por testes (TDD)

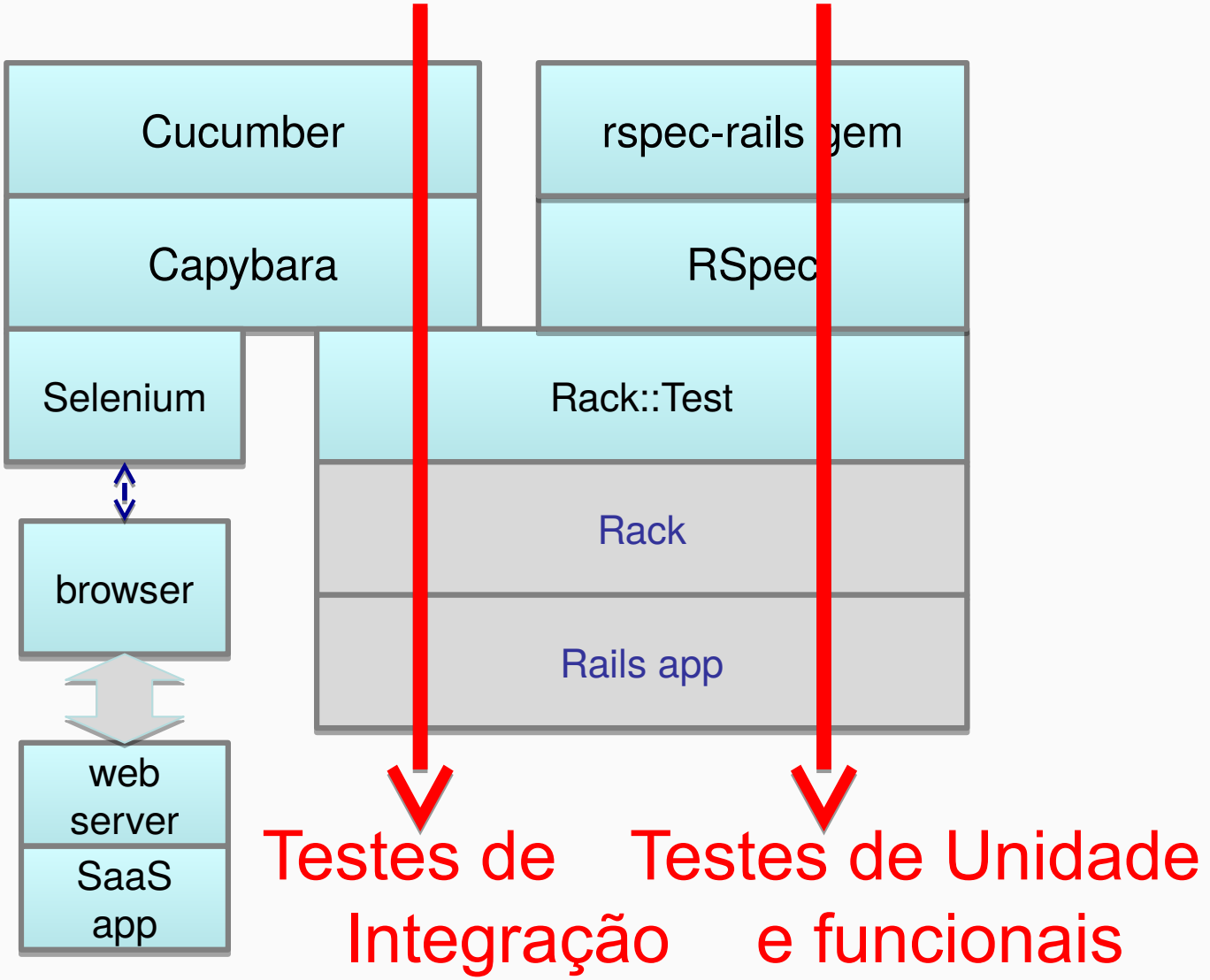
- cada *definição de passo* para uma nova história pode precisar que se desenvolva novo código
- TDD advoca que: escreva os testes de unidade & funcionais primeiro, antes de escrever o código
- ou seja, escreva testes para **o código que você gostaria de ter**

Cucumber & RSpec

- Cucumber descreve o *comportamento* com as funcionalidades & cenários (projeto guiado pelo comportamento)
- RSpec testa os módulos individuais que contribuem com esses comportamentos (desenvolvimento guiado por testes)



PILHA DE TESTES



FIRST, TDD e RSpec

TESTES DE UNIDADE DEVEM SER FIRST

F ast (rápido)

I ndepenent (independente)

R epeatable (repetível)

S elf-checking (autoverificável)

T imely (oportuno)

TESTES DE UNIDADE DEVEM SER FIRST

Rápido rodar (um subconjunto dos) testes deve ser rápido (já que você vai fazer isso o tempo todo)

TESTES DE UNIDADE DEVEM SER FIRST

Rápido rodar (um subconjunto dos) testes deve ser rápido (já que você vai fazer isso o tempo todo)

Independente testes não devem depender uns dos outros, você deve poder rodá-los quaisquer testes em qualquer ordem

TESTES DE UNIDADE DEVEM SER FIRST

Rápido rodar (um subconjunto dos) testes deve ser rápido (já que você vai fazer isso o tempo todo)

Independente testes não devem depender uns dos outros, você deve poder rodá-los quaisquer testes em qualquer ordem

Repetível N execuções sempre devem produzir o mesmo resultado (para ajudar a isolar bugs e permitir a automação)

TESTES DE UNIDADE DEVEM SER FIRST

Rápido rodar (um subconjunto dos) testes deve ser rápido (já que você vai fazer isso o tempo todo)

Independente testes não devem depender uns dos outros, você deve poder rodá-los quaisquer testes em qualquer ordem

Repetível N execuções sempre devem produzir o mesmo resultado (para ajudar a isolar bugs e permitir a automação)

Autoverificável testes devem poder detectar por si mesmos se foram bem sucedidos (não deve haver uma pessoa para verificar os resultados)

TESTES DE UNIDADE DEVEM SER FIRST

Rápido rodar (um subconjunto dos) testes deve ser rápido (já que você vai fazer isso o tempo todo)

Independente testes não devem depender uns dos outros, você deve poder rodá-los quaisquer testes em qualquer ordem

Repetível N execuções sempre devem produzir o mesmo resultado (para ajudar a isolar bugs e permitir a automação)

Autoverificável testes devem poder detectar por si mesmos se foram bem sucedidos (não deve haver uma pessoa para verificar os resultados)

Oportuno escrito quase que ao mesmo tempo que o código que será testado (com TDD, é escrito antes do código!)

RSPEC

- Linguagem específica de domínio (DSL) para testes.
  - DSL são pequenas linguagens de programação que simplificam uma tarefa, mas que normalmente são menos generalizáveis
  - Exs: migrações, expressões regulares, SQL
- Testes em RSpec são chamados de specs ou exemplos
- Para rodar os testes em um arquivo: `rspec <arquivo>`
- Ou melhor: use `guard/autotest`

EXEMPLO DE RSPEC

```ruby
expect { k += 1.05 }.to change { k }.by( a_value_within(0.1).of(1.0) )

expect { s = "barn" }.to change { s }
  .from( a_string_matching(/foo/) )
  .to( a_string_matching(/bar/) )

expect(["barn", 2.45]).to contain_exactly(
  a_value_within(0.1).of(2.5),
  a_string_starting_with("bar")
)

expect(["barn", "food", 2.45]).to end_with(
  a_string_matching("foo"),
  a_value > 2
)

expect(["barn", 2.45]).to include( a_string_starting_with("bar") )

expect(:a => "food", :b => "good").to include(:a => a_string_matching(/foo/))
```

https://github.com/rspec/rspec-expectations

```ruby
require 'ruby_intro.rb'

describe BookInStock do
  it "should be defined" do
    expect { BookInStock }.not_to raise_error
  end

describe 'getters and setters' do
  before(:each)  { @book = BookInStock.new('isbn1', 33.8) }
  it 'sets ISBN' do
    expect(@book.isbn).to eq('isbn1')
  end
  it 'sets price' do
    expect(@book.price).to eq(33.8)
  end
  it 'can change ISBN' do
    @book.isbn = 'isbn2'
    expect(@book.isbn).to eq('isbn2')
  end
  it 'can change price' do
    @book.price = 300.0
    expect(@book.price).to eq(300.0)
  end
end
```

```
expect { lambda }.to(assertion)
expect(expression).to(assertion)
```

PERGUNTA

Quais tipos de código podem ser testados de forma **R**epetível e **I**ndependente?

1. Código que depende de aleatoriedade (ex: misturar um baralho de cartas
2. Código que depende do horário do dia (ex: faz backup todo domingo à meia-noite)

Resposta:

- só (1)
- só (2)
- tanto (1) quanto (2)
- nem (1) nem (2)