# Web sistemų testavimas

Saulius Grigaitis

# Testavimas

- Testavimas – techninis tyrinėjimas, kurio tikslas yra pateikti informaciją apie produkto arba paslaugos kokybę [Wik08]
- Testavimas atskleidžia produkto arba paslaugos atitikimą specifikacijai.
- Kuriant programinę įrangą nepavyksta išvengti defektų, nes žmonės daro klaidas
- Ankščiau atrastus defektus pašalinti kainuoja pigiau.
- Netrivialiems produktams ar paslaugos neįmanoma rasti visų klaidų

# “Unit” testavimas

- “Unit” testavimas – mažiausių sistemos vienetų(metodų, funkcijų) testavimas
- Tikslas:
  - Izoliavus mažiausius sistemos vienetus užtikrinti korektišką jų veikimą
- Uždaviniai:
  - Sugalvoti testuojamo vieneto interfeisą.
  - Izoliuoti testuojamą vienetą.
  - Parašyti testus, kurių kodas tikrina testuojamų vienetų korektišką veikimą(dažniausiai tikrinama imituojant realų testuojamo vieneto naudojimą)
- “Baltos dėžės” principas
- Beveik visais atvejais testuoja pats programuotojas

# “Unit” testavimas – praktiniai privalumai

- Efektyvi priemonė nuo streso ir depresijos :)
- Gerina kodo kokybę
- Leidžia mažomis sąnaudomis pertvarkyti kodą
- Leidžia rasti daugumą “techninių” klaidų
- Palengvina integracijos procesą ir testavimą
- Priverčia programuotoją sukurti “gyvą” dokumentaciją
- “TDD/BDD”(angl. “Test Driven Development”/”Behavior driven development”), naudojamas “ekstremalaus programavimo” metodikoje.

# RSpec

- “Elgsena paremtas testavimas”(angl. “Behaviour Driven Development” - “BDD”) Ruby kalba
- Du įrankiai:
  - Specifikacijų (angl. “spec”) įrankis – elgsenai apibūdinti objekto lygyje
  - Pasakojimų (angl. “story”) įrankis – elgsenai apibūdinti programos lygyje
- Testavimas baltos dėžės principu

# Rspec

- Tinka testuoti “Ruby” kalba parašytą kodą, taigi tiek “Ruby on Rails”, tiek kitus WEB karkasus “Ruby” kalba, pvz. “Merb”
- Labai natūrali “DSL” kalba skirta nusakyti sistemos elgseną
- Pateikia klases “MVC” komponentų testavimui
- Pateikia “MVC” komponentų ir jų testų generatorius

# Rspec – specifikavimas

Programuotojas: “Describe an account when it is first created”

Užsakovas: “It should have balance of $0”

```
describe Account, “ when first created” do
  it “should have a balance of $0” do
    @account = Account.new
    @account.balance.should eql(0)
  end
end
```

# RSpec – specifikacijos generavimas

- Komanda “rake spec:doc”
- Specifikacija iš pavyzdžio ankstesnėje skaidėje:

```
Account, when first created
- should have a balance of $0
```

# RSpec – testo struktūra

```
describe Thing, “ when it is in some context” do
  before(:all) do
  end

  before(:each) do
  end

  it “should do stuff” do
  end

  it “should do more stuff” do

  after() do
  end

  after(:all) do
  end
end
```

# Rspec – tikėjimosi metodai

- Du tikėjimos metodai:
  - “should(matcher = nil)” - tikimasi, kad nustatymo metodas(angl. “matcher”) praeis.
  - “should_not(matcher = nil)” - tikimasi, kad nustatymo metodas(angl. “matcher”) nepraeis.
- Pavyzdžiai:
  - @student.should have(11).books
  - @student.should_not be_lazy
  - @lecturer.should be_inspiring

# RSpec – nustatymo metodai I

- [].should be_empty => [].empty? # praeina
- [].should_not be_instance_of(Array) => 3.instance_of?(Fixnum) #praeina
- {:a => “A”}.should have_key(:b) => {:a => “A”}.has_key?(:b) #nepraeina
- @true.should be_true #preina, jei @true != false

# RSpec – nustatymo metodai II

- Lambda { employee.develop_great_new_social_network_app }.should change(employee, :title).from(“Mail Clerk”).to(“CEO”)
- Lambda { nil[] }.should raise_error(NoMethodError)

# RSpec – savas nustatymo metodas I

```
Class Lazy
  def initialize(expected)
    @expected = expected
  end

  def matches?(target)
    @target = target
    EXAMS_SESSION_START_DATE - Time.now <
1.week && target.completed_tasks = 0
  end
...
```

# RSpec – savas nustatymo metodas II

```
...
  def failure_message
    “expected #{@target.inspect} to be lazy”
  end
  def negative_failure_message
    “expected #{@target.inspect} not to be lazy”
  end
end

def be_lazy(expected)
  Lazy.new(expected)
end
```

# Rspec – “Ruby on Rails” MVC kodo ir testų generavimas

- “rspec_model” - sugeneruoja esybės modelį, testų karkasą, duombazės migracijos failą, testinių duomenų failą
- “rspec_controller” - sugeneruoja esybės kontrolerį, maršrutizavimo konfigūraciją, testus jai bei kontroleriui.
- “rspec_scaffold” - panaudoja “rspec_model” ir “rspec_controller”, bei sugeneruoja vaizdus ir testus jiems

# RSpec - klausimai

?

# Šaltinių sąrašas


- http://wtr.rubyforge.org/documentation.html
- http://code.google.com/p/firewatir/w/list
- http://selenium-core.openqa.org/documentation.jsp
- http://selenium-ide.openqa.org/documentation.jsp
- http://selenium-rc.openqa.org/documentation.jsp
- http://selenium-grid.openqa.org/how_it_works.html
- http://rspec.info/examples.html
- [Wik08] http://en.wikipedia.org/wiki/Software_testing