Implementing TDD at work

a case study in a Ruby on Rails dev team

Gonzalo Bulnes Guilpain

May 30, 2013

DynLang Chile Meetup

1. focus on what's important
2. prefer continuous over disruptive
3. don't do more than you can

focus on what's important

programming languages

allow people to communicate

focus on what's important

with people

that's why we adopt *agile working methods*

"*Individuals and interactions* over processes and tools"

– The Agile Manifesto

so, *don't* focus on the **code** tonight

because you'll find easily the RSpec matchers here when you need them

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

focus on the **mindset** and

why we write some tests and ignore others

focus on people, the people you work with

prefer continuous over disruptive

we're talking about *team management* here

prefer continuous over disruptive

Test to make your co-workers happier

write specs as *documentation*

so they can get introduced painlessly to your code

```
$ rspec --format documentation
```

```
$ echo “--format documentation” > .rspec
```

prefer continuous over disruptive

start on your own

prefer continuous over disruptive

let your co-workers decide *how* and *when* to start,
they'll probably start right after you *if it's not much extra work*

“Intrinsic motivation relies on *autonomy*, mastery and purpose.”

– Daniel Pink, Drive

prefer continuous over disruptive

don't be affraid to start more than once

because you skiped bold announcements,

you co-workers may not even have noticed you started twice

prefer continuous over disruptive

then show them, don't tell them

well documented code is worth a thousand good reasons

but tell them how progressively you started

don't do more than you can

"Think big, start small. [...] For long-term change, experiment immediately."

– Eric Ries, The Lean Startup

don't do more than you can

Hypotesis: if they feel happier with your code when it's tested, devs will embrace TDD driven by *purpose*.

Remember: “your co-workers [...] *if it's not much extra work*”

At this point, more extra work than time means no intrinsic motivation at all.

I use to start with models specs

- simple spec examples
- easy to maintain
- learning included (factories)
- extensible (validations, methods)

```ruby
# specs/models/video_spec.rb
require 'spec_helper'

describe Video do

  # attributes

  # associations

  # validations

  # methods

end
```

```ruby
# specs/models/video_spec.rb
require 'spec_helper'

describe Video do

  # attributes

  it “has a width” do
    should respond_to :width
  end

  # ...

end
```

```ruby
# specs/models/video_spec.rb
require 'spec_helper'

describe Video do

  # validations

  it “has a valid Factory” do
    FactoryGirl.build(:video).should be_valid
  end

  # ...

end
```

```ruby
# specs/factories/videos.rb
FactoryGirl.define do |f|

  factory :video do
  end
end
```

```ruby
# specs/models/video_spec.rb
require 'spec_helper'

describe Video do

  # validations

  it “requires width presence” do
    FactoryGirl.build(:video, width: '').should_not be_valid
  end

  # ...

end
```

```ruby
# specs/factories/videos.rb
FactoryGirl.define do |f|

  factory :video do
    width 240
  end
end
```

```ruby
# specs/factories/videos.rb
FactoryGirl.define do |f|

  factory :video do
    width 240
  end

  factory :invalid_video, class: :video do
  end
end
```

```ruby
# specs/models/video_spec.rb
describe Video do

  # validations

  it “has an invalid Factory” do
    FactoryGirl.build(:invalid_video).should_not be_valid
  end
end
```

```ruby
# specs/models/video_spec.rb
require 'spec_helper'

describe Video do

  # associations

  it “belongs to an article” do
    should belong_to :article
  end

  # ...

end
```

```ruby
# specs/factories/videos.rb
FactoryGirl.define do |f|

  factory :video do
    article
    width 240
  end

  # ...

end
```

```
gonzalo@cassis:~/dev/blog[add-videos]$ rspec spec/models/video_spec.rb

Video
  has an height
  has an invalid factory
  requires width presence
  requires its height to be an integer
  has a width
  requires height presence
  requires its width to be an integer
  requires file name presence
  belongs to an article
  has a valid factory
  has a file name

Finished in 0.07867 seconds
11 examples, 0 failures

Randomized with seed 49893
```

keep specs simple

as soon as associations get tricky, skip factories

then establish common conventions

with your fellow devs

if you're here, you were right

don't do more than you can

Hypotesis: green test suites are addictive, once they started testing, devs main drive to TDD is *mastery*.

Tip: you'll keep moving one step ahead of your co-workers aiming at showing them *the TDD way*

routing specs quickly

```ruby
# specs/routing/reservations_routing_spec.rb
require "spec_helper"

describe ReservationsController do
  describe "routing" do

    it "routes to #index" do
      get("/reservations").should route_to("reservations#index")
    end

  end
end
```

```ruby
# specs/routing/reservations_routing_spec.rb
require "spec_helper"

describe ReservationsController do
  describe "routing" do

    # ...

    it "routes to #new" do
      get("/room/1/reservar").should \
                        route_to("reservations#new", :room_id => "1")
    end

  end
end
```

lead to controllers specs

- scaffold is a good draft
- DRY sweeties
- deeper factories usage

```ruby
require 'spec_helper'

describe VideosController do

  # This should return the minimal set of attributes bla bla bla...
  def valid_attributes
    { width: 235 }
  end

  describe "GET index" do
    it "assigns all places as @videos" do
      video = Video.create! valid_attributes
      get :index, {}
      assigns(:videos).should eq([video])
    end

    it "renders the administration layout" do
      get :index
      response.should render_template 'layouts/administration'
    end
  end

  # ...
end
```

```ruby
require 'spec_helper'

describe VideosController do

  def valid_attributes
    FactoryGirl.attributes_for(:video).stringify_keys!
  end

  def invalid_attributes
    FactoryGirl.attributes_for(:invalid_video).stringify_keys!
  end

  # ...
end
```

```ruby
require 'spec_helper'

describe VideosController do
  # see https://github.com/thoughtbot/factory_girl
    FactoryGirl.build_stubbed(:video)
  # see https://www.relishapp.com/rspec/rspec_mocks
    Video.any_instance.stub(:save).and_return(false)
  # ...
end
```

follow the path you prefer

e.g. I like to have views helpers well documented

because it's easy to forget they exist

```
gonzalo@cassis:~/dev/blog[add-videos]$ rspec spec/helpers/videos_helper_spec.rb

VideosHelper
  #results_title
    when params[:category] is not set
      raises RuntimeError
    when @videos is not set
      raises RuntimeError
    when params[:category] and @videos are set
      returns a title containing the serach results count
      and params[:year] is set
        returns a title with the search results count and year information

Finished in 0.07537 seconds
4 examples, 0 failures

Randomized with seed 21384
```

conclusion

1. do TDD for your team
2. experiments sometimes fail
3. be patient

Some reading, about BDD

by David Chelimsky (the RSpec author) The RSpec Book

One last thing:

be ready to answer questions

innovation is about people, and people usually get questions

This talk is free.

This document and its sources can be found at

https://github.com/gonzalo-bulnes

Thank you.