Maintaining balance while reducing duplication

David Chelimsky

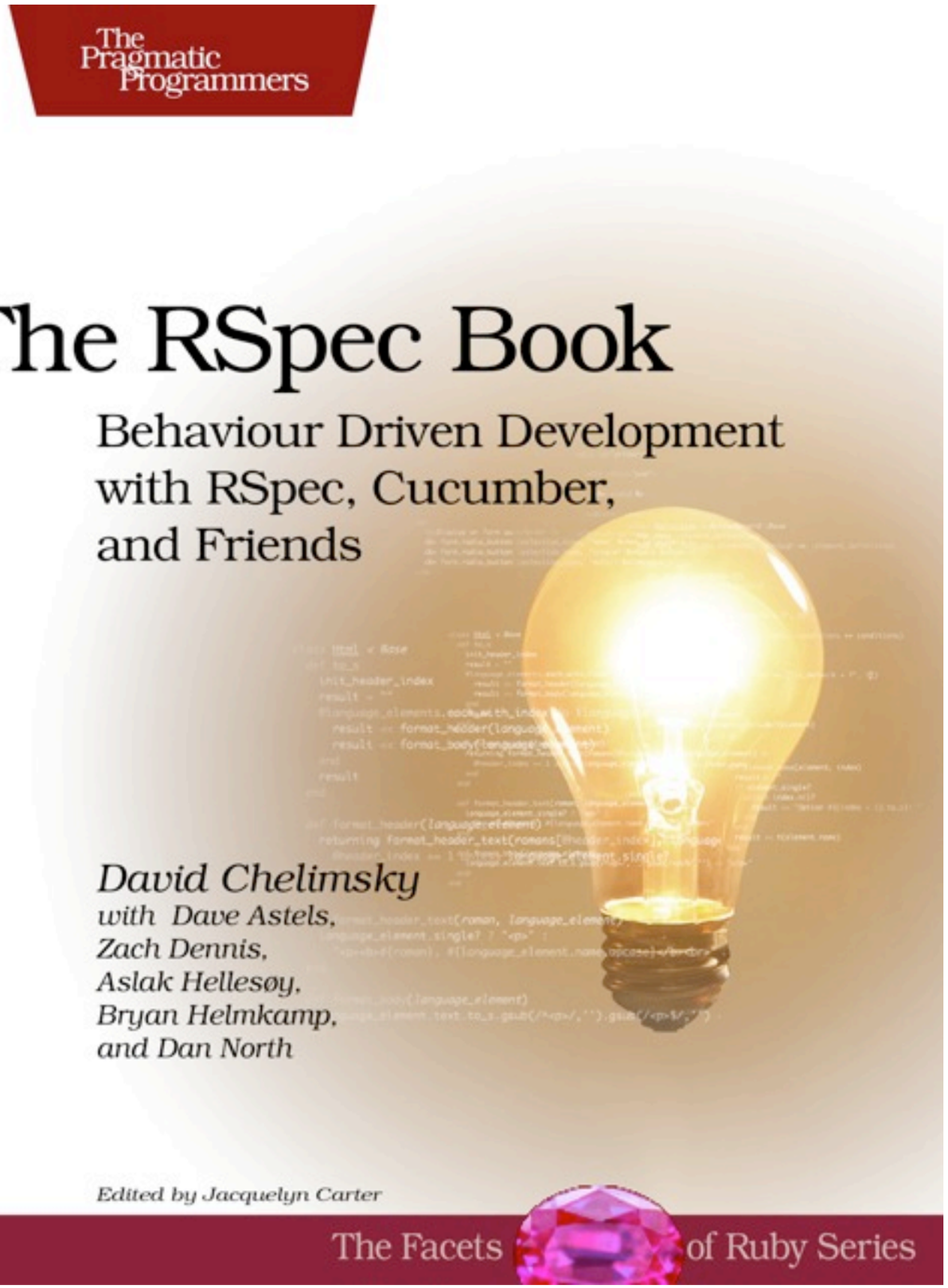
http://drw.com

DRW TRADING GROUP

About Us | Trading | Technology

http://rubygems.org/gems/rspec

http://pragprog.com/titles/achbd/the-rspec-book



This talk is not about RSpec

This talk **is** about

duplication

Duplication is bad

“Duplicate code is the root of all evil in software design.”

Code smell:
Duplicated Code

The DRY principle

Don’t
Repeat
Yourself

Custom Matchers : Shouldn't they follow DRY principle ?

```ruby
describe WidgetsController do
  describe "POST create" do
    it "returns 201 (created)"
      post :create, :widget => {:name => 'ACME Widget'}
      response.should be_created
    end
  end
end

describe WidgetsController do
  describe "POST create" do
    it "creates the widget and returns 201 (created)" do
      post :create, :widget => {:name => 'ACME Widget'}
      widget = Widget.find_by_name('ACME Widget')
      response.should have_created_resource(widget)
    end
  end
end
```

```ruby
describe WidgetsController do
  describe "POST create" do
    it "returns 201 (created)"
      post :create, :widget => {:name => 'ACME Widget'}
      response.should be_created
    end
  end
end

describe WidgetsController do
  describe "POST create" do
    it "creates the widget and returns 201 (created)" do
      post :create, :widget => {:name => 'ACME Widget'}
      widget = Widget.find_by_name('ACME Widget')
      response.should have_created_resource(widget)
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end

  failure_message_for_should do |response|
    @message
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end

  failure_message_for_should do |response|
    @message
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end

  failure_message_for_should do |response|
    @message
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end

  failure_message_for_should do |response|
    @message
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    if !be_created.matches?(response)
      @message = be_created.failure_message_for_should(response)
    elsif !be_nil.matches?(resource)
      @message = "expected the resource not to be nil"
    end
  end

  failure_message_for_should do |response|
    @message
  end
end
```

Every time you
reduce duplication
you
increase coupling
by introducing
new dependencies

Dependencies
have to be
managed

SOLID Principles

Single Responsibility

Open/Closed

Liskov Substitution

Interface Segregation

Dependency Inversion

The Dependency Inversion Principle

The Dependency Inversion Principle

Depend on **abstractions**

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == "201"
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == "201"
  end

  failure_message_for_should do |response|
    if response.code != "201"
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code == CREATED
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code == CREATED
  end

  failure_message_for_should do |response|
    if response.code != CREATED
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    "expected the response code to be 201 (created) but was #{response.code}"
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      "expected the response code to be 201 (created) but was #{response.code}"
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      "expected the resource not to be nil"
    end
  end
end
```

Code is easier to understand when it operates at a consistent level of abstraction

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      "expected the resource not to be nil"
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      FailureMessages.expected_non_nil_resource
    end
  end
end
```

```ruby
RSpec::Matchers.define :be_created do
  match do |response|
    response.code.created?
  end

  failure_message_for_should do |response|
    FailureMessages.expected_response_code_for(:created).got(response.code)
  end
end

RSpec::Matchers.define :have_created_resource do |resource|
  match do |response|
    resource && response.code.created?
  end

  failure_message_for_should do |response|
    if !response.code.created?
      FailureMessages.expected_response_code_for(:created).got(response.code)
    else
      FailureMessages.expected_non_nil_resource
    end
  end
end
```

The DRY principle

Don’t
Repeat
Yourself

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system

```
"/allocations/{sourceType}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
```

```
"/allocations/{sourceType}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
```

```
ALLOCATIONS = "allocations"

ALLOCATIONS_FOR_TYPE = ALLOCATIONS +
  "/{sourceType}/{year}/{month}"

ALLOCATIONS_FOR_INSTANCE = ALLOCATIONS +
  "/{sourceType}/{sourceId}/{year}/{month}"

CREATE_ALLOCATION = ALLOCATIONS +
  "/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
```

```
ALLOCATIONS = "allocations"
SOURCE_TYPE = "/{sourceType}"


ALLOCATIONS_FOR_TYPE = ALLOCATIONS + SOURCE_TYPE
  "/{year}/{month}"

ALLOCATIONS_FOR_INSTANCE = ALLOCATIONS + SOURCE_TYPE
  "/{sourceId}/{year}/{month}"

CREATE_ALLOCATION = ALLOCATIONS + SOURCE_TYPE
  "/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"
```

```
ALLOCATIONS     = "allocations"
SOURCE_TYPE     = "/{sourceType}"
SOURCE_ID       = "/{sourceId}"
TARGET_TYPE     = "/{targetType}"
TARGET_ID       = "/{targetId}"
EFFECTIVE_DATE = "/{year}/{month}"
PERCENTAGE      = "/{percentage}"


ALLOCATIONS_FOR_TYPE     = ALLOCATIONS + SOURCE_TYPE + EFFECTIVE_DATE

ALLOCATIONS_FOR_INSTANCE = ALLOCATIONS + SOURCE_TYPE + SOURCE_ID + EFFECTIVE_DATE

CREATE_ALLOCATION         = ALLOCATIONS + SOURCE_TYPE + SOURCE_ID +
  TARGET_TYPE + TARGET_ID + EFFECTIVE_DATE + PERCENTAGE
```

```
DELIMITERS           = %w[/ { }]
EFFECTIVE_DATE       = DELIMITERS[0..1].join + "year" + DELIMITERS[2] +
  DELIMITERS[0..1].join + "month" + DELIMITERS[2]
ALLOCATIONS          = "allocations"
SOURCE               = "source"
TARGET               = "target"
ID                   = "Id"
TYPE                 = "Type"
PERCENTAGE           = "percentage"
SOURCE_TYPE          = DELIMITERS[0..1].join + SOURCE + TYPE + DELIMITERS[2]
SOURCE_ID            = DELIMITERS[0..1].join + SOURCE + ID + DELIMITERS[2]
TARGET_TYPE          = DELIMITERS[0..1].join + TARGET + TYPE + DELIMITERS[2]
TARGET_ID            = DELIMITERS[0..1].join + TARGET + ID + DELIMITERS[2]
ALLOCATION_SOURCE    = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET    = TARGET_TYPE + TARGET_ID
ALLOCATIONS_FOR_TYPE      = ALLOCATIONS + SOURCE_TYPE +
    DELIMITERS[0] + EFFECTIVE_DATE
ALLOCATIONS_FOR_INSTANCE = ALLOCATIONS + ALLOCATION_SOURCE +
    DELIMITERS[0] + EFFECTIVE_DATE
CREATE_ALLOCATION    = ALLOCATIONS + ALLOCATION_SOURCE +
    ALLOCATION_TARGET + DELIMITERS[0] + EFFECTIVE_DATE +
    DELIMITERS[0..1].join + PERCENTAGE + DELIMITERS[2]
```

```
"/allocations/developer/37/project/42/2010/10/87"

DELIMITERS           = %w[/ { }]
EFFECTIVE_DATE       = DELIMITERS[0..1].join + "year" + DELIMITERS[2] +
  DELIMITERS[0..1].join + "month" + DELIMITERS[2]
ALLOCATIONS          = "allocations"
SOURCE               = "source"
TARGET               = "target"
ID                   = "Id"
TYPE                 = "Type"
PERCENTAGE           = "percentage"
SOURCE_TYPE          = DELIMITERS[0..1].join + SOURCE + TYPE + DELIMITERS[2]
SOURCE_ID            = DELIMITERS[0..1].join + SOURCE + ID + DELIMITERS[2]
TARGET_TYPE          = DELIMITERS[0..1].join + TARGET + TYPE + DELIMITERS[2]
TARGET_ID            = DELIMITERS[0..1].join + TARGET + ID + DELIMITERS[2]
ALLOCATION_SOURCE    = SOURCE_TYPE + SOURCE_ID
ALLOCATION_TARGET    = TARGET_TYPE + TARGET_ID
ALLOCATIONS_FOR_TYPE      = ALLOCATIONS + SOURCE_TYPE +
    DELIMITERS[0] + EFFECTIVE_DATE
ALLOCATIONS_FOR_INSTANCE = ALLOCATIONS + ALLOCATION_SOURCE +
    DELIMITERS[0] + EFFECTIVE_DATE
CREATE_ALLOCATION    = ALLOCATIONS + ALLOCATION_SOURCE +
    ALLOCATION_TARGET + DELIMITERS[0] + EFFECTIVE_DATE +
    DELIMITERS[0..1].join + PERCENTAGE + DELIMITERS[2]
```

"/allocations/developer/37/project/42/2010/10/87"

"/allocations/{sourceType}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{year}/{month}"
"/allocations/{sourceType}/{sourceId}/{targetType}/{targetId}/{year}/{month}/{percentage}"

DRY does not mean
“don’t type the same
characters twice”

duplicated concepts

isolation of change

duplication

is

evil

not all

duplication

is

evil

How do I know if this particular duplication is problematic?

Purpose/Intention

Risk

Have to change in more than one place?

Might forget to
change in other place!

Might forget to repeat
it in the third place!

Structural/Functional

```ruby
class Person
  def primary_zip_code
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.zip_code if primary
  end
end
```

```ruby
class Person
  def primary_zip_code
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.zip_code if primary
  end

  def primary_phone_number
  end
end
```

```ruby
class Person
  def primary_zip_code
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.zip_code if primary
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

```ruby
class Person
  def primary_zip_code
    main = addresses.detect {|a| a.primary?}
    main ||= addresses.first
    main if main
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

```ruby
class Person
  def primary_zip_code
    main = addresses.detect {|a| a.primary?} || addresses.first
    main.zip_code if main
  end

  def primary_phone_number
    primary = addresses.detect {|a| a.primary?}
    primary ||= addresses.first
    primary.phone_number if primary
  end
end
```

```ruby
class Person
  def primary_zip_code
    primary_address.zip_code
  end

  def primary_phone_number
    primary_address.phone_number
  end

  def primary_address
    addresses.detect {|a| a.primary?} ||
                      addresses.first ||
                      Address.new
  end
end
```

Same name, different meaning

browser

Percentages:

50.0

30.0

20.0

report

percentage
0.5
0.3
0.2

Same meaning, different API

```ruby
class Line
  attr_accessor :start_point, :end_point, :length

  def initialize(start_point, end_point, length)
    @start_point, @end_point, @length = start_point, end_point, length
  end
end
```

```ruby
class Line
  attr_accessor :start_point, :end_point, :length

  def initialize(start_point, end_point, length)
    @start_point, @end_point, @length = start_point, end_point, length
  end
end


start_point = Point.new(0,0)
end_point = Point.new(37,42)
length = Math.hypot(end_point.x - start_point.x, end_point.y - start_point.y)
line = Line.new(start_point, end_point, length)

line.length
```

```ruby
class Line
  attr_reader :start_point, :end_point

  def initialize(start_point, end_point)
    @start_point, @end_point = start_point, end_point
  end

  def length
    Math.hypot(end_point.x - start_point.x, end_point.y - start_point.y)
  end
end
```

```ruby
class Line
  attr_reader :start_point, :end_point

  def initialize(start_point, end_point)
    @start_point, @end_point = start_point, end_point
  end

  def length
    Math.hypot(end_point.x - start_point.x, end_point.y - start_point.y)
  end
end
```

```ruby
line = Line.new(Point.new(0,0), Point.new(37,42))

line.length
```

Specs

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        it "returns nil"
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        it "returns nil" do
          person = Person.new
          person.addresses << Address.new(:zipcode => nil)
          person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        it "returns nil" do
          person = Person.new
          person.addresses << Address.new(:zipcode => nil)
          person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        before do
          @person = Person.new.tap do |person|
            person.addresses << Address.new(:zipcode => nil)
          end
        end

        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        before do
          @person = Person.new.tap do |person|
            person.addresses << Address.new(:zipcode => nil)
          end
        end

        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        before do
          @person = Person.new.tap do |person|
            person.addresses << Address.new(:zipcode => nil)
          end
        end

        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      context "with no zipcode" do
        before do
          @person = Person.new.tap do |person|
            person.addresses << Address.new(:zipcode => nil)
          end
        end

        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person = Person.new.tap do |person|
          person.addresses << Address.new
        end
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person = Person.new.tap do |person|
          person.addresses << Address.new
        end
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person = Person.new.tap do |person|
          person.addresses << Address.new
        end
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person = Person.new.tap do |person|
          person.addresses << Address.new
        end
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

```ruby
describe Person do
  before do
    @person = Person.new
  end
  describe "#distance_to_event" do
    context "with one address" do
      before do
        @person.addresses << Address.new
      end
      context "with no zipcode" do
        before do
          @person.addresses.first.zipcode = nil
        end
        it "returns nil" do
          @person.distance_to_event(Event.new).should be(nil)
        end
      end
    end
  end
end
```

Less duplication

often means

more indirection

Indirection can make it **harder** to **reason** about **code**

```ruby
describe Person do
  describe "when first created" do
    it "has no friends" do
      Person.new.should have(:no).friends
    end
  end
end
```

```ruby
describe Person do
  describe "when first created" do
    it "has no friends" do
      Person.new.should have(:no).friends
    end
  end
end
```

```
Person
  when first created
    has no friends
```

```ruby
describe Person do
  describe "when first created" do
    it "has no friends" do
      Person.new.should have(:no).friends
    end
  end
end
```

```
Person
  when first created
    has no friends
```

```ruby
describe Person do
  describe "when first created" do
    it { should have(:no).friends }
  end
end
```

```ruby
describe Person do
  describe "when first created" do
    it "has no friends" do
      Person.new.should have(:no).friends
    end
  end
end
```

```
Person
  when first created
    has no friends
```

```ruby
describe Person do
  describe "when first created" do
    it { should have(:no).friends }
  end
end
```

```
Person
  when first created
    should have 0 friends
```

```ruby
describe Calculator do
  describe "#add" do
    it "returns the sum of the operands provided" do
      calculator = Calculator.new
      calculator.add(2,3).should eq(5)
    end
  end
end
```

```ruby
describe Calculator do
  describe "#add" do
    it "returns the sum of the operands provided" do
      calculator = Calculator.new
      calculator.add(2,3).should eq(5)
    end
  end
end
```

```
Calculator
  #add
    returns the sum of the operands provided
```

```ruby
describe Calculator do
  describe "#add" do
    it "returns the sum of the operands provided" do
      calculator = Calculator.new
      calculator.add(2,3).should eq(5)
    end
  end
end
```

```
Calculator
  #add
    returns the sum of the operands provided
```

```ruby
describe Calculator do
  subject { Calculator.new.add(2,3) }
  it { should eq(5) }
end
```

```ruby
describe Calculator do
  describe "#add" do
    it "returns the sum of the operands provided" do
      calculator = Calculator.new
      calculator.add(2,3).should eq(5)
    end
  end
end
```

```
Calculator
  #add
    returns the sum of the operands provided
```

```ruby
describe Calculator do
  subject { Calculator.new.add(2,3) }
  it { should eq(5) }
end
```

```
Calculator
  should eq 5
```

Helpful tools can be
harmful
when used in the
wrong context

Rails Controllers

```ruby
class WidgetsController < ApplicationController
  def index
    @widgets = Widget.all
  end

  def show
    @widget = Widget.find(params[:id])
  end

  def new
    @widget = Widget.new
  end

  def edit
    @widget = Widget.find(params[:id])
  end

  def create
    @widget = Widget.new(params[:widget])
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end
```

```
def update
  @widget = Widget.find(params[:id])

  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget = Widget.find(params[:id])
  @widget.destroy
  redirect_to(widgets_url)
end
end
```

```ruby
class WidgetsController < ApplicationController
  def show
    assign_widget
  end

  def edit
    assign_widget
  end

  def update
    assign_widget
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    assign_widget
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end
end
```

```ruby
class WidgetsController < ApplicationController
  def show
    assign_widget
  end

  def edit
    assign_widget
  end

  def update
    assign_widget
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    assign_widget
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end
end
```

Before filters!

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :only => [:show, :edit, :update, :destroy]

  def show; end

  def edit; end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end

end
```

```ruby
def index
  @widgets = Widget.all
end

def show; end

def new
  @widget = Widget.new
end

def edit; end

def create
  @widget = Widget.new(params[:widget])

  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end

def update
  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget.destroy
  redirect_to(widgets_url)
end
```

```ruby
def index
  @widgets = Widget.all
end

def show; end

def new
  @widget = Widget.new
end

def edit; end

def create
  @widget = Widget.new(params[:widget])

  if @widget.save
    redirect_to(@widget, :notice => 'Widget was successfully created.')
  else
    render :action => "new"
  end
end

def update
  if @widget.update_attributes(params[:widget])
    redirect_to(@widget, :notice => 'Widget was successfully updated.')
  else
    render :action => "edit"
  end
end

def destroy
  @widget.destroy
  redirect_to(widgets_url)
end
```

```ruby
  def new
    @widget = Widget.new
  end

  def create
    @widget = Widget.new(params[:widget])
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  end

private

  def assign_widget
    @widget = Widget.find(params[:id])
  end
```

```ruby
  def new

  end

  def create

    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  end

  private

    def assign_widget
      @widget = params[:id] ? Widget.find(params[:id]) :
                              Widget.new(params[:widget])
    end
```

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => :index

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

```ruby
class WidgetsController < ApplicationController
  before_filter :assign_widget, :except => index

  def index
    @widgets = Widget.all
  end

  def show; end
  def new; end
  def edit; end

  def create
    if @widget.save
      redirect_to(@widget, :notice => 'Widget was successfully created.')
    else
      render :action => "new"
    end
  end

  def update
    if @widget.update_attributes(params[:widget])
      redirect_to(@widget, :notice => 'Widget was successfully updated.')
    else
      render :action => "edit"
    end
  end

  def destroy
    @widget.destroy
    redirect_to(widgets_url)
  end

private

  def assign_widget
    @widget = params[:id] ? Widget.find(params[:id]) : Widget.new(params[:widget])
  end

end
```

In Summary

Many problems in software can be traced back to duplication, but ...

... not all duplication
is inherently evil

Duplication comes in many forms

Duplication appears
for **several reasons**

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system

Every time you
reduce duplication
you
increase coupling
by introducing
new dependencies

Depend on
abstractions

Context matters

Questions?

@dchelimsky

http://blog.davidchelimsky.net/

http://relishapp.com/rspec/

http://www.drwtrading.com/

http://www.pragprog.com/the-pragmatic-programmer

http://pragprog.com/titles/achbd/the-rspec-book

http://www.amazon.com/
Refactoring-Improving-Design-Existing-Code/
dp/0201485672

http://www.amazon.com/
Software-Development-Principles-Patterns-Practices/
dp/0135974445