DOMAIN-DRIVEN RAILS

THE EVOLUTION OF RAILS

from a Theory of Constraints perspective

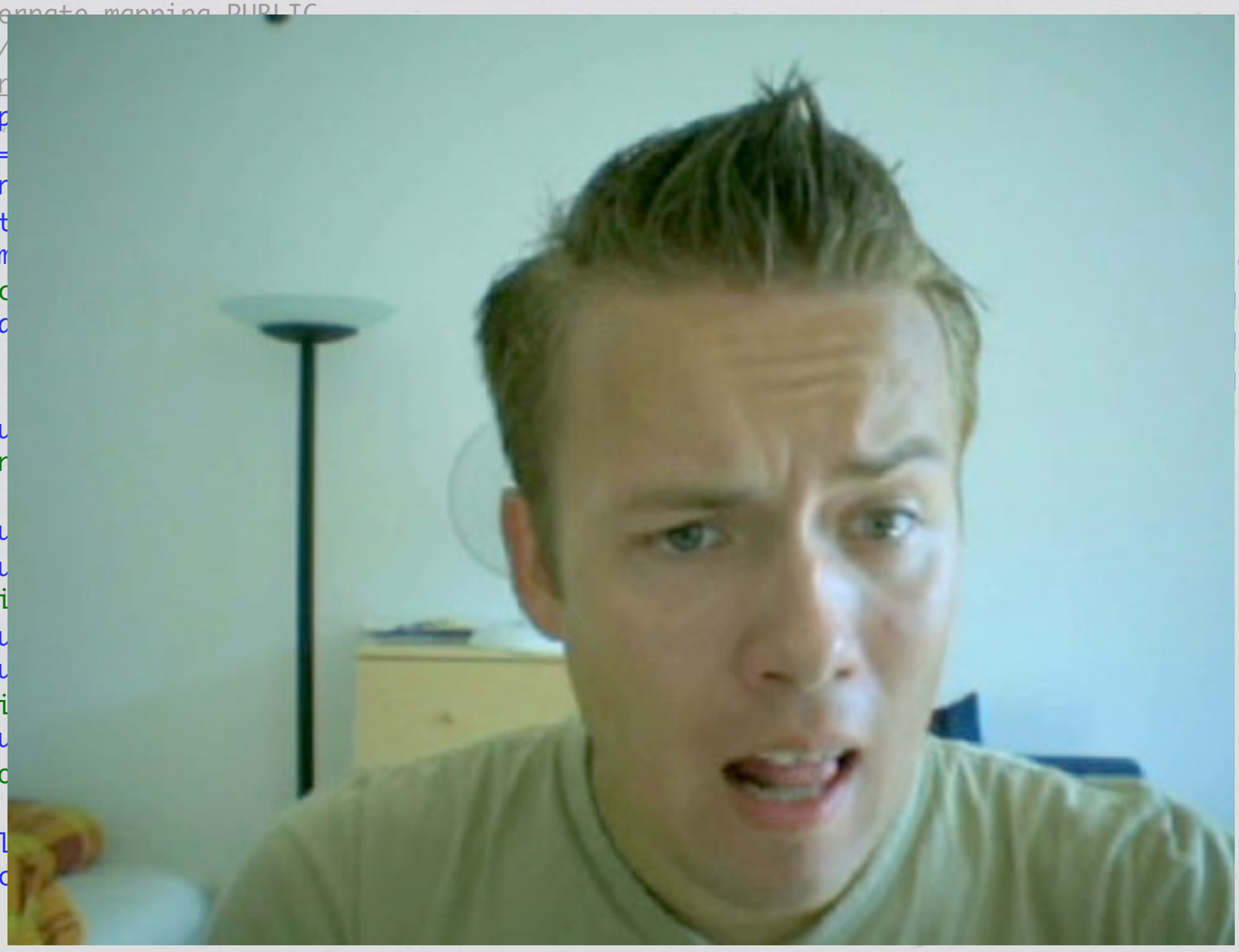
DHH

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="org.applabs.base.User" table="USER">
    <id column="USER_ID" name="id" type="java.lang.Long">
      <generator class="org.hibernate.id.TableHiLoGenerator">
        <param name="table">idgen</param>
<param name="column">NEXT</param>
      </generator>
    </id>
    <property column="USER_NAME" name="userName" type="java.lang.String"/>
<property column="USER_PASSWORD" name="userPassword"
type="java.lang.String"/>
...
<property column="CREATED_BY" name="createdBy" type="java.lang.Double"/>
<property column="MODIFICATION_DATE" length="4" name="modificationDate"
type="java.util.Date"/>
<property column="MODIFIED_BY" name="modifiedBy" type="java.lang.Double"/>
<property column="DELETE_DATE" length="4" name="deleteDate"
type="java.util.Date"/>
<property column="DELETED_BY" name="deletedBy" type="java.lang.Double"/>
<set name="properties" lazy="true" inverse="true" cascade="all-delete-
orphan">
      <key column="USER_ID" />
<one-to-many class="org.applabs.base.UserProp" />
    </set>
  </class>
</hibernate-mapping>
```

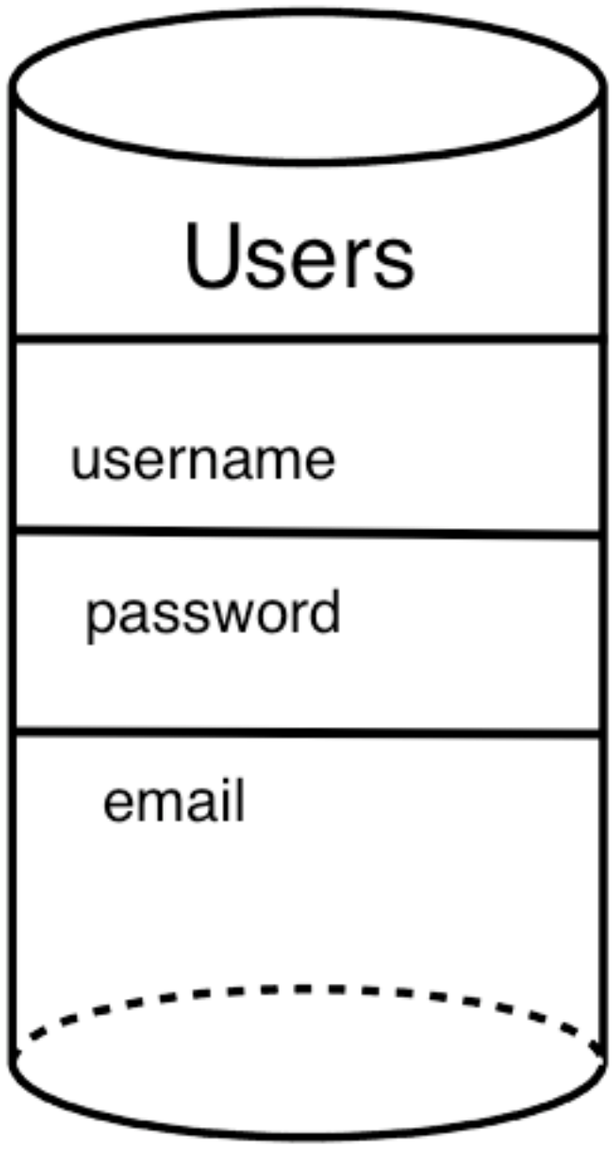
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
  <class name="org.applabs.base.User" table="USER">
    <id column="USER_ID" name="id" type="java.lang.Long">
      <generator class="org.hibernate.id.TableHiLoGenerator">
        <param name="table">idgen</param>
<param name="column">NEXT</param>
      </generator>
    </id>
    <property column="USER_NAME" name="userName" type="java.lang.String"/>
<property column="USER_PASSWORD" name="userPassword"
type="java.lang.String"/>
...
<property column="CREATED_BY" name="createdBy" type="java.lang.Double"/>
<property column="MODIFICATION_DATE" length="4" name="modificationDate"
type="java.util.Date"/>
<property column="MODIFIED_BY" name="modifiedBy" type="java.lang.Double"/>
<property column="DELETE_DATE" length="4" name="deleteDate"
type="java.util.Date"/>
<property column="DELETED_BY" name="deletedBy" type="java.lang.Double"/>
<set name="properties" lazy="true" inverse="true" cascade="all-delete-
orphan">
      <key column="USER_ID" />
<one-to-many class="org.applabs.base.UserProp" />
    </set>
  </class>
</hibernate-mapping>
```

```
public static class User {
  private String userName;
  private String userPassword;
  private String userEmail;
  ...
}
```

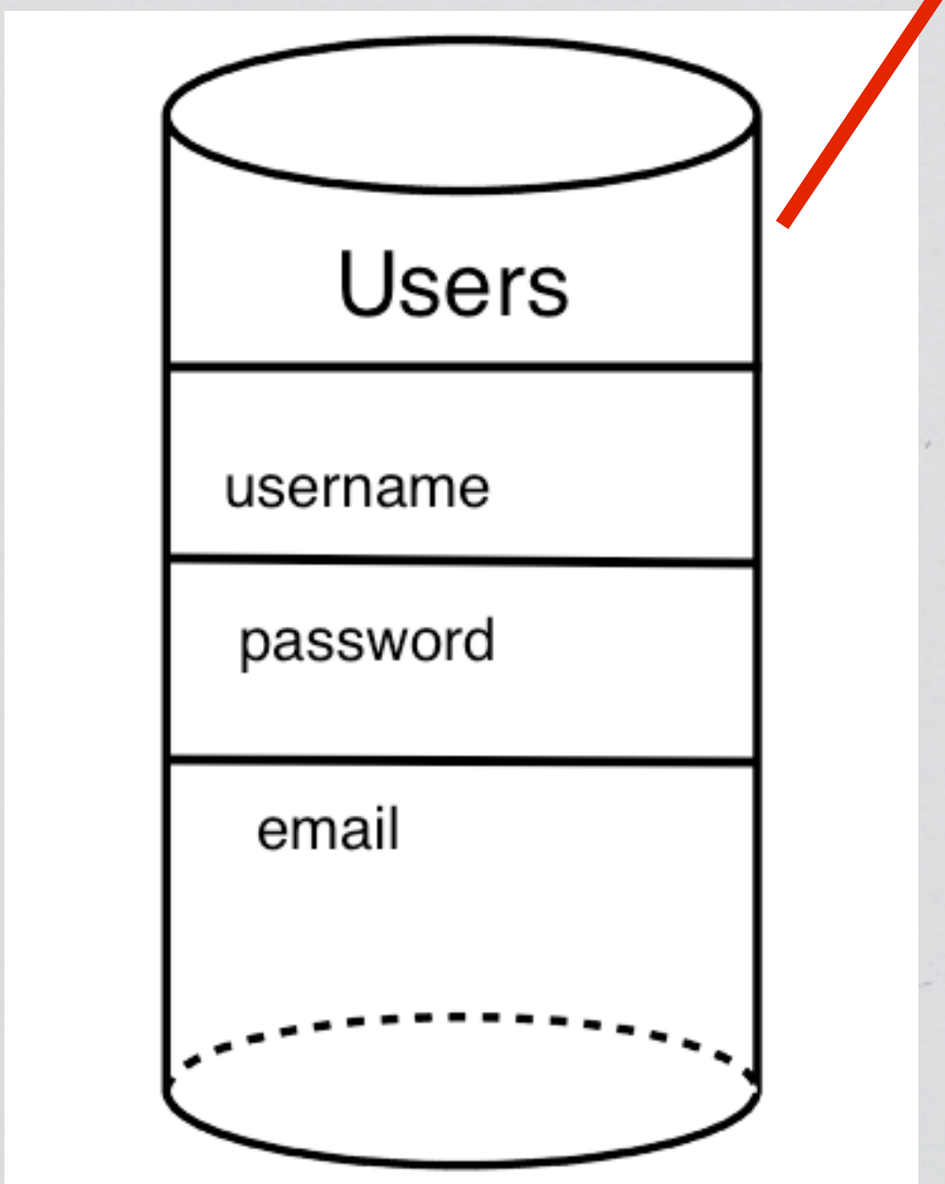
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/
"http://hiberr
<hibernate-map
  <class name=
    <id columr
      <generat
        <param
<param name="c
      </genera
    </id>
    <property
<property colu
type="java.lar
...
<property colu
<property colu
type="java.uti
<property colu
<property colu
type="java.uti
<property colu
<set name="pro
orphan">
      <key col
<one-to-many c
    </set>
  </class>
</hibernate-mapping>
```



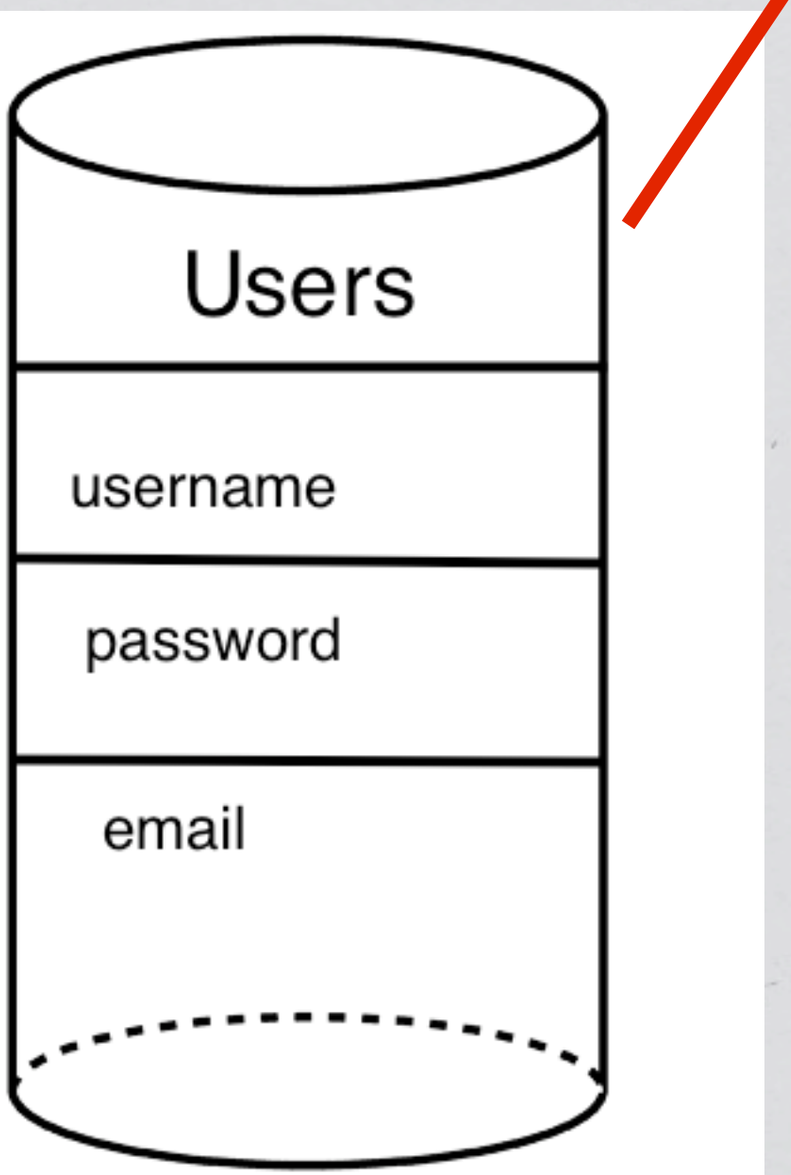
```
ass User {
 userName;
 userPassword;
 userEmail;
```


Users
username
password
email

```ruby
class User < ActiveRecord::Base
end
```



```ruby
class User < ActiveRecord::Base
end
```



```
class User < ActiveRecord::Base
end
```

Constraint: Immediate development

Constraint: Immediate development

convention
configuration

Constraint: Immediate development

convention
configuration

Constraint: Immediate development

convention
configuration


```
Terminal — bash — 86×10
~/blahblah:$ ./script/generate model User username:string password:string email:string
      exists  app/models/
      exists  test/unit/
      exists  test/fixtures/
      create  app/models/user.rb
      create  test/unit/user_test.rb
      create  test/fixtures/users.yml
      create  db/migrate
      create  db/migrate/20091001151750_create_users.rb
~/blahblah:$
```

New constraint:
Day to day maintenance

New constraint:
Day to day maintenance

New constraint: Day to day maintenance

RSpec, shoulda, bacon, context, etc

New constraint: Day to day maintenance

RSpec, shoulda, bacon, context, etc

metaprogramming

static code generation

New constraint: Day to day maintenance

RSpec, shoulda, bacon, context, etc

metaprogramming

static code generation

"Constraints are liberating"

Skinny Controller, Fat Model

```
class UsersController
  def index
    @users = User.find :all,
      :conditions => ['last_visit_at >= ?', 7.days.ago]
  end
end
```

Skinny Controller, Fat Model

```
class UsersController
  def index
    @users = User.find :all,
      :conditions => ['last_visit_at >= ?', 7.days.ago]
  end
end
```

Skinny Controller, Fat Model

```
class UsersController
  def index
    @users = User.recent
  end
end

class User
  def self.recent(num_days = 7)
    find :all,
      :conditions => ['last_visit_at >= ?', num_days.days.ago]
  end
end
```

SOMETHING INTERESTING HAPPENED

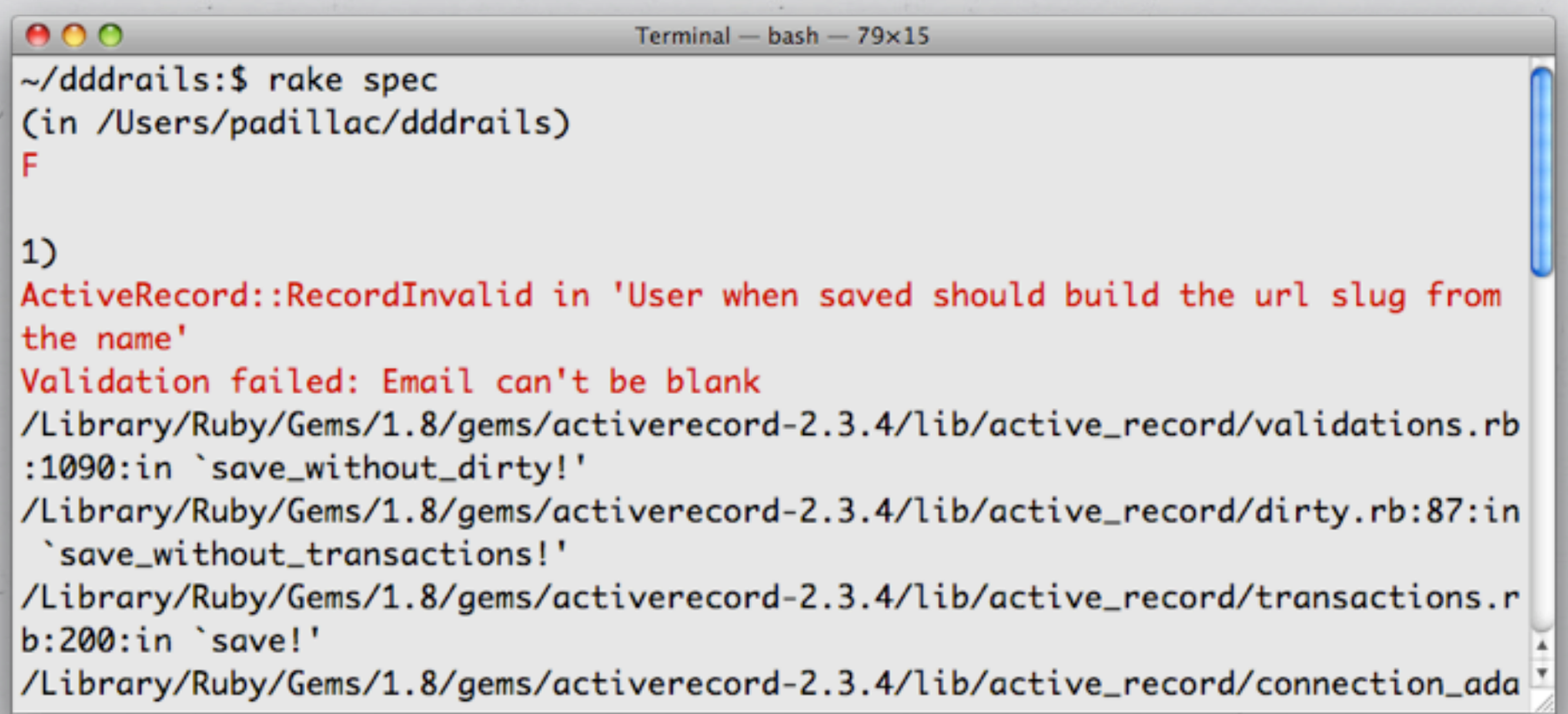
```ruby
class UsersController
  def create
    @user = User.new params[:user]
    if @user.save
      UserMailer.deliver_signup @user
    else
      render :action => "new"
    end
  end
end
```

```
class UsersController
  resource_controller
end

class User
  after_create :deliver_signup_email

  def deliver_signup_email
    UserMailer.deliver_signup self
  end
end
```

WHAT HAPPENED NEXT?

```
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = User.create! :name => "Pat Maddox"
    user.url_slug.should == "pat-maddox"
  end
end
```

```ruby
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = User.create! :name => "Pat Maddox"
    user.url_slug.should == "pat-maddox"
  end
end
```



```
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = User.new :name => "Pat Maddox"
    user.save false
    user.url_slug.should == "pat-maddox"
  end
end
```

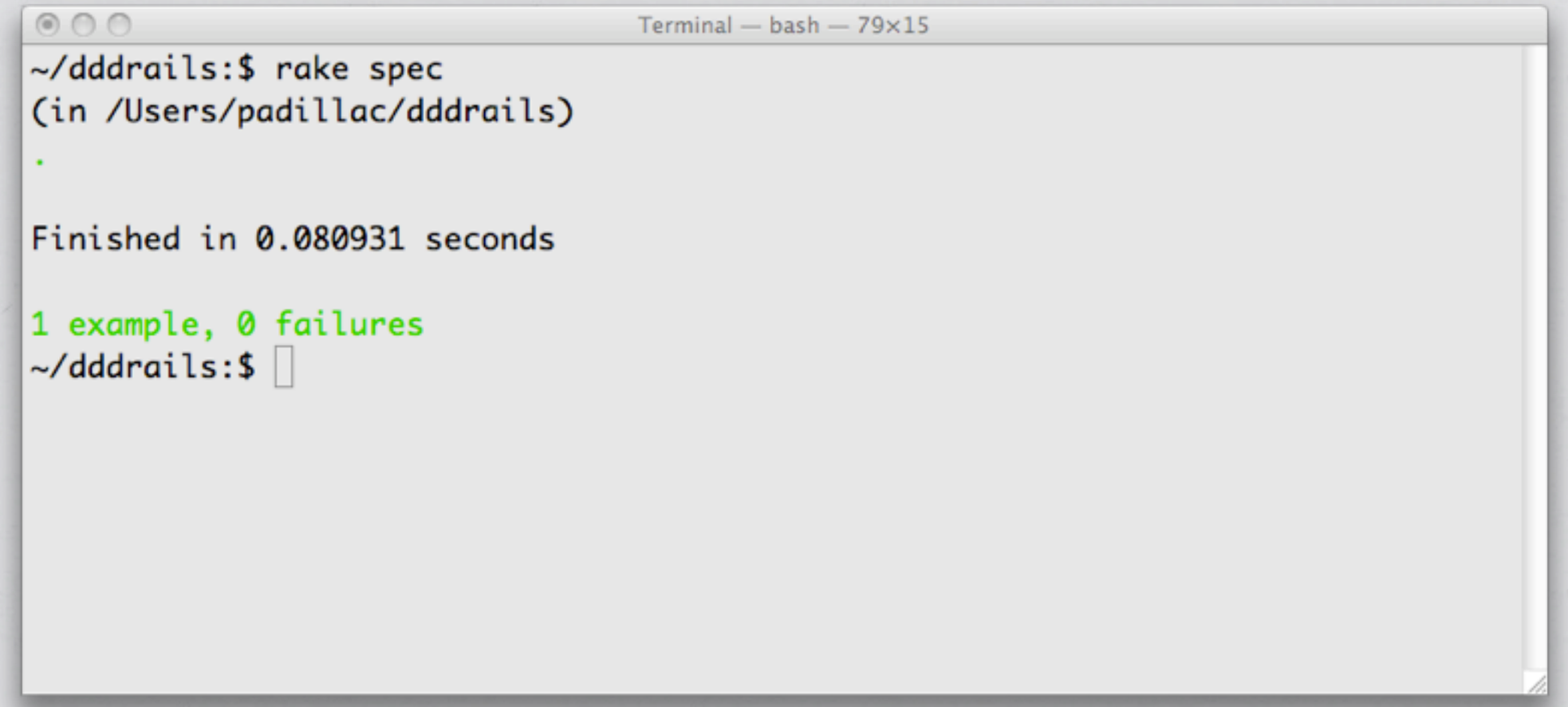
```ruby
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = User.new :name => "Pat Maddox"
    user.save false
    user.url_slug.should == "pat-maddox"
  end
end
```

Terminal — bash — 79×15

```
~/dddrails:$ rake spec
(in /Users/padillac/dddrails)
F

1)
NoMethodError in 'User when saved should build the url slug from the name'
You have a nil object when you didn't expect it!
The error occurred while evaluating nil.gsub
/Users/padillac/dddrails/app/models/user_mailer.rb:3:in `signup'
/Library/Ruby/Gems/1.8/gems/actionmailer-2.3.4/lib/action_mailer/base.rb:459:in
 `__send__'
/Library/Ruby/Gems/1.8/gems/actionmailer-2.3.4/lib/action_mailer/base.rb:459:in
 `create!'
/Library/Ruby/Gems/1.8/gems/actionmailer-2.3.4/lib/action_mailer/base.rb:452:in
 `initialize'
```

```
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = create_user :name => "Pat Maddox" # or Factory(:user, ...)
    user.url_slug.should == "pat-maddox"
  end
end
```

```ruby
describe User, "when saved" do
  it "should build the url slug from the name" do
    user = create_user :name => "Pat Maddox" # or Factory(:user, ...)
    user.url_slug.should == "pat-maddox"
  end
end
```



A SIGN OF THINGS TO COME?

WHO CARES? JUST SHIP!!!

My boss asks “can you add invites?”

I say, “sure, just give me a few days”

```ruby
class InvitesController
  def show
    invite = Invite.find_by_token! params[:id]
    session[:invite_token] = invite.token
    redirect_to new_user_path
  end
end

class UsersController
  resource_controller
  before_filter :set_invite_token_from_session, :only => :create

  def set_invite_token_from_session
    if session[:invite_token]
      params[:user][:invite_token] = session[:invite_token]
    end
  end
end
```

Eric, my QA guy, says, "dude what the heck? Invited users are getting email confirmation requests"

So?

Eric, my QA guy, says, "dude what the heck? Invited users are getting email confirmation requests"

So?

We already know that their email address is good because they received the invitation!

Oh...that makes sense

```ruby
class User
  after_create :deliver_signup_email, :unless => :invite_token?

  def deliver_signup_email
    UserMailer.deliver_signup self
  end
end
```

Eric: We still need to verify the email address if they sign up with a different address than the invitation was sent to!

Eric: We still need to verify the email address if they sign up with a different address than the invitation was sent to!

This is getting complex. Why don't we pair on Cucumber?

```
Feature: Accept invitation
  Scenario: Sign up with same email address as the invitation
    Given an invitation sent to "pat.maddox@gmail.com"
    When I accept the invitation
    And sign up with the email address "pat.maddox@gmail.com"
    Then I should not receive a confirmation email

  Scenario: Sign up with different address than the invitation
    Given an invitation sent to "pat.maddox@gmail.com"
    When I accept the invitation
    And sign up with the email address "pmaddox@goldstar.com"
    Then I should receive a confirmation email
```

```ruby
class User
  belongs_to :invite
  after_create :deliver_signup_email, :if => :needs_confirmation?

  def deliver_signup_email
    UserMailer.deliver_signup self
  end

  def needs_confirmation?
    invite.blank? || invite.email_address != email_address
  end
end
```

AR callbacks are over-used

AR callbacks are over-used

* Create strange dependencies

AR callbacks are over-used

* Create strange dependencies
* Make tests brittle and slow

AR callbacks are over-used

* Create strange dependencies
* Make tests brittle and slow
* Breeding ground for confusing code

AR callbacks are over-used

* Create strange dependencies
* Make tests brittle and slow
* Breeding ground for confusing code
* Hide important domain concepts

Alternatives to callbacks

Alternatives to callbacks

* Move the logic back into the controller

Alternatives to callbacks

* Move the logic back into the controller
* Services

Alternatives to callbacks

* Move the logic back into the controller
* Services
* Command objects

Alternatives to callbacks

* Move the logic back into the controller
* Services
* Command objects
* Event handlers

Example: Using a service

```
class SignupUserService
  def signup(user_options={})
    user = User.new user_options
    UserMailer.deliver_signup(user) if user.save
    user
  end
end

class AcceptInvitationService
  def accept(invitation, user_options={})
    user = User.new user_options
    if user.save && user.email != invitation.recipient_email
      UserMailer.deliver_signup(user) if user.save
    end
    user
  end
end
```

2005
SKINNY CONTROLLER, FAT MODEL

2009

USE YOUR LAYERS WISELY

Why does it matter?

Why does it matter?

* Applications are becoming more complex

Why does it matter?

* Applications are becoming more complex
* Application domains are becoming more complex

Why does it matter?

* Applications are becoming more complex
* Application domains are becoming more complex
* Developer happiness!

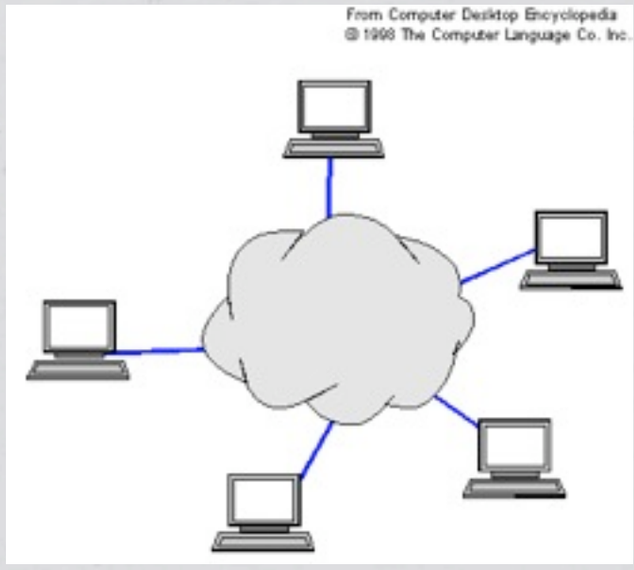
Why does it matter?

* Applications are becoming more complex
* Application domains are becoming more complex
* Developer happiness!
* $$ for the business

Constraint: Long-term strategy

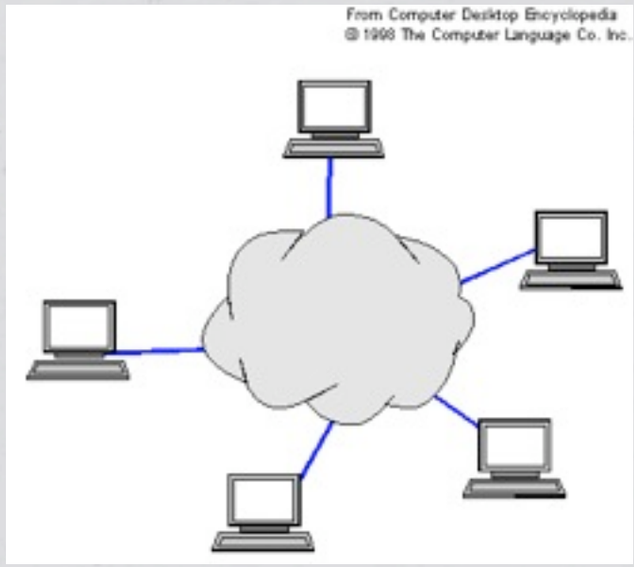
Constraint: Long-term strategy

Constraint: Long-term strategy



Constraint: Long-term strategy

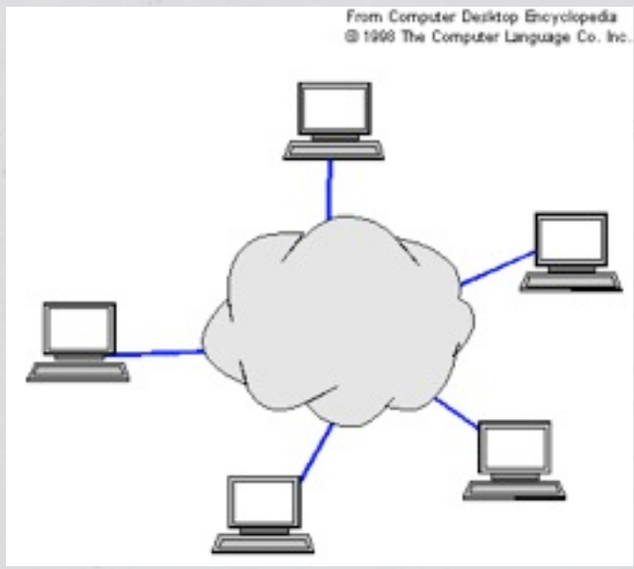
Cucumber



Constraint: Long-term strategy

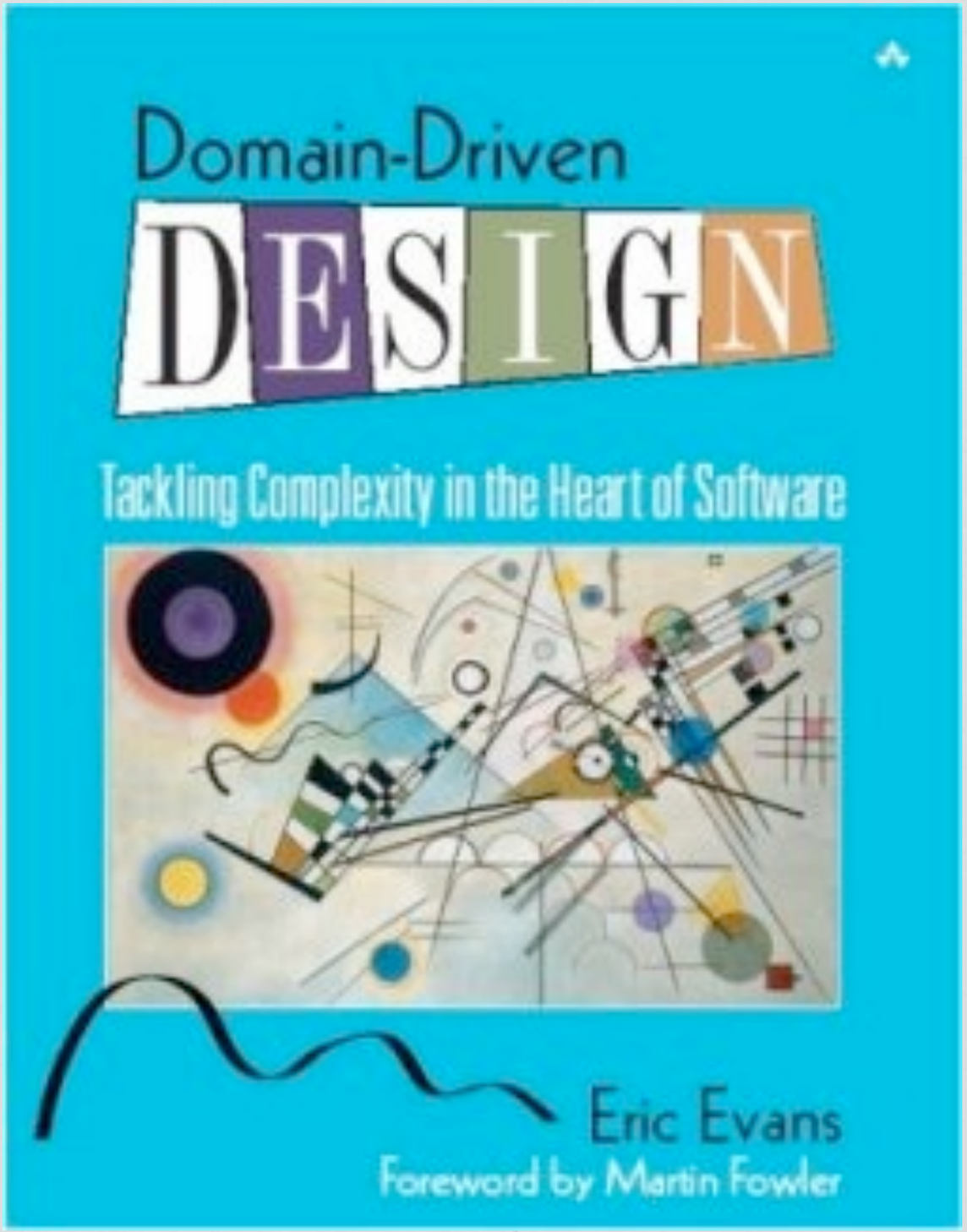
Observational & Switchboard

Cucumber



Focus on the core domain

* The secret sauce of your application
* The area to focus on when doing BDD


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler

Where to go from here

http://www.twitter.com/patmaddox

http://www.patmaddox.com

pat.maddox@gmail.com

Where to go from here

* Read “Domain-Driven Design”

---

http://www.twitter.com/patmaddox
http://www.patmaddox.com
pat.maddox@gmail.com

Where to go from here

* Read “Domain-Driven Design”
* Challenge assumptions and “best practices”

---

http://www.twitter.com/patmaddox
http://www.patmaddox.com
pat.maddox@gmail.com

Where to go from here

* Read “Domain-Driven Design”
* Challenge assumptions and “best practices”
* Take a holistic approach to software craftsmanship

---

http://www.twitter.com/patmaddox
http://www.patmaddox.com
pat.maddox@gmail.com