RubyFocus

Efficient Rails Test-Driven Development

Wolfram Arnold
www.rubyfocus.biz

In collaboration with:
Sarah Allen, BlazingCloud.net
marakana.com

Class Outline

RubyFocus

Week #1

- The economics of testing
- Testing in layers, design patterns
- Toolbox: RSpec with Rails
- RSpec & Models

Week #2

- A culture of testing: Why TDD? How to TDD?
- Testing & Data Dependencies
- Toolbox: Fixtures, Factories, Mocks & Stubs

Class Outline

RubyFocus

Week #3

- Controller testing
- View, Helper, Routes Testing
- How much is enough? How much is too much?

Week #4

- Refactoring code & tests, custom matchers
- API Testing
- Remote data setup
- Cucumber for API testing & documentation

Class Outline

RubyFocus

Week #5

- Integration Testing—when and how?
- Toolbox: Cucumber, Selenium & Friends
- Page Object Pattern

Week #6

- TDD—what next?
- BDD, Agile Process, XP → VDD: Value-Driven Development
- Advanced/Special Requests

What you can expect

RubyFocus

Presentation, Examples

Answers to questions

A range of material from current development practice

Homework

Fluidity & adaptability

Fun

It works best, when...

RubyFocus

- Active participation
- Try something new
- Focus
- Team Effort
  - Pairing
  - Utilizing the resources in class: TA's, participants
- Discussions in public
  - Class Q&A
  - Mailing list

Efficient Rails Test-Driven Development

Why “efficient” and “testing”?

“Testing takes too much time.”

“It's more efficient to test later.”

“Testing is the responsibility of QA, not developers.”

“It's not practical to test X.”

- Tests break when data changes.
- Tests break when design changes.

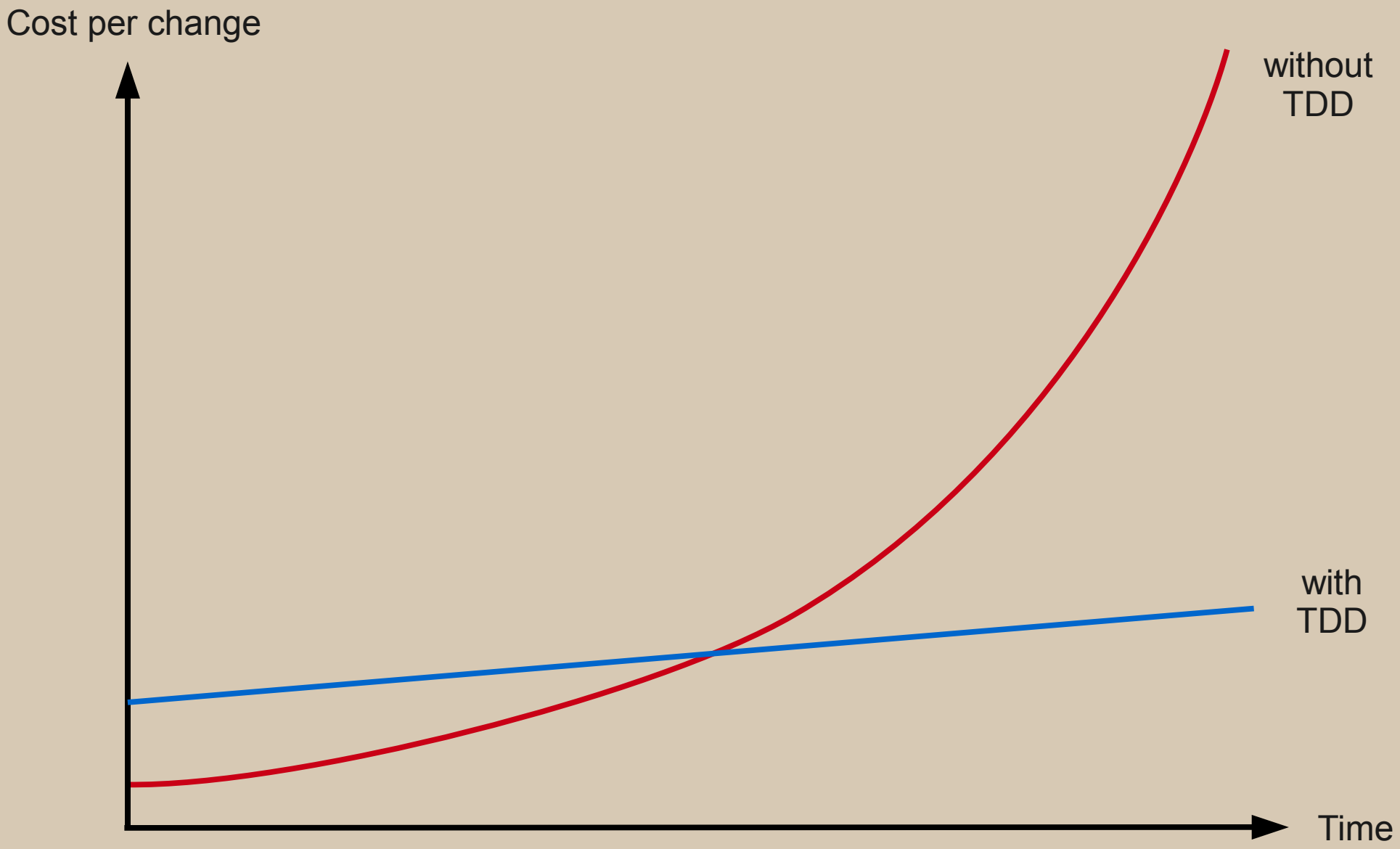
The Role of Testing

RubyFocus

Development without tests...

- fails to empower developers to efficiently take responsibility for quality of the code delivered
- makes collaboration harder
- build narrow silos of expertise
- instills fear & resistance to change
- makes documentation a chore
- stops being efficient very soon

TDD: Keeping cost of change low

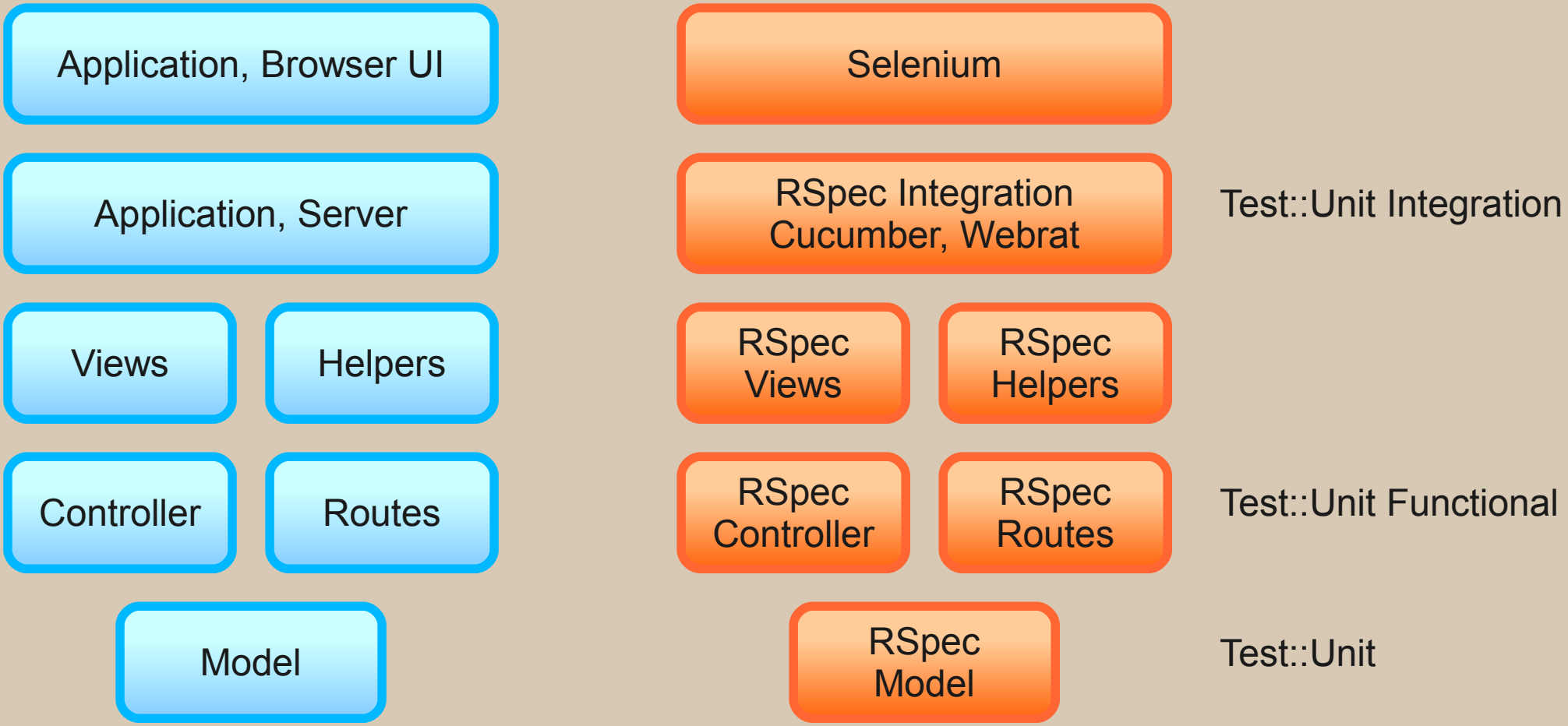


Why?

RubyFocus

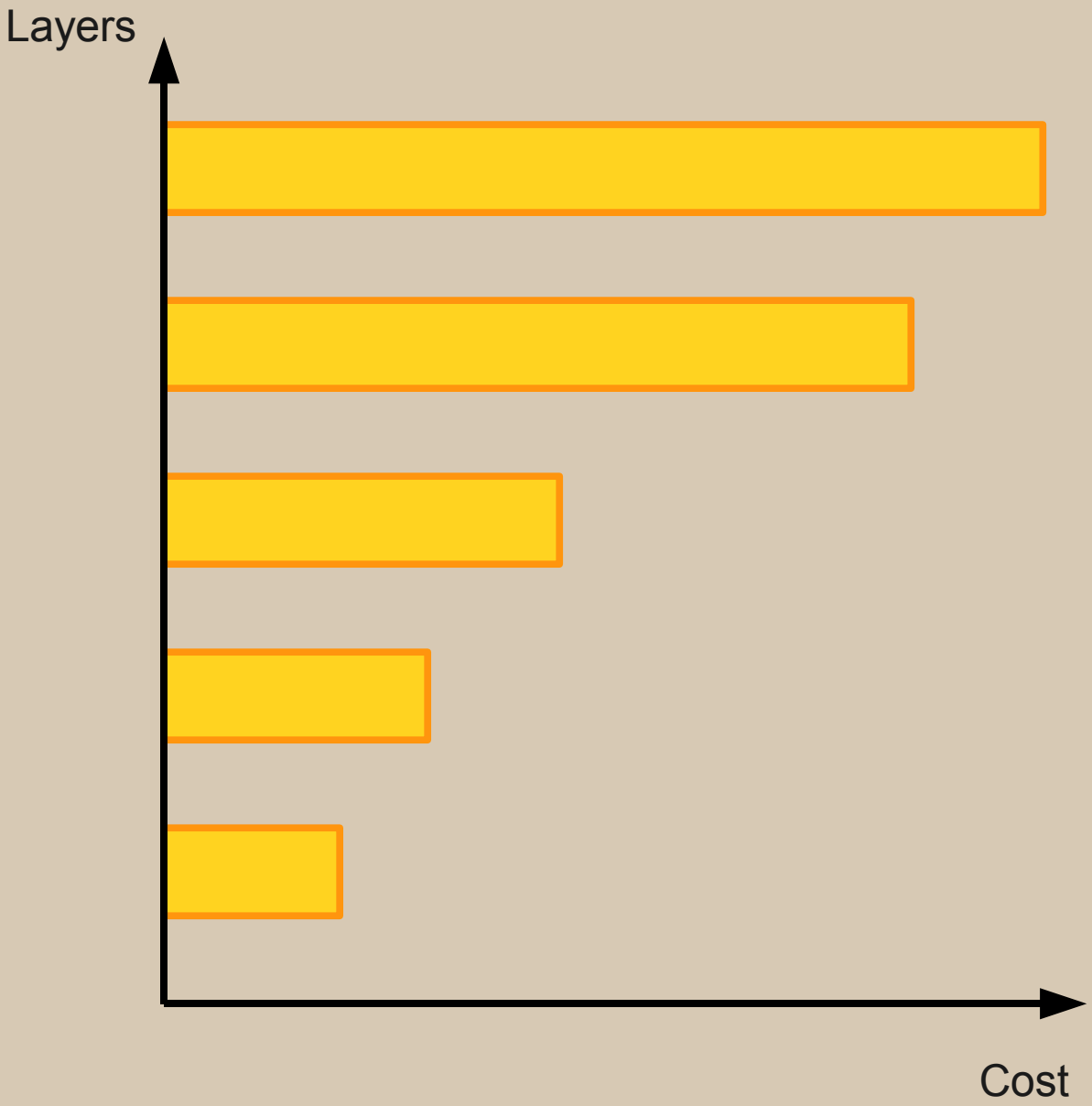
- Non-TDD
  - Accumulates “technical debt” unchecked
  - Removal of technical debt carries risk
    - The more technical debt, the higher the risk
  - Existing technical debt attracts more technical debt
    - Like compound interest
    - People are most likely to do what others did before them
    - To break the pattern heroic discipline & coordination required

Testing in Layers

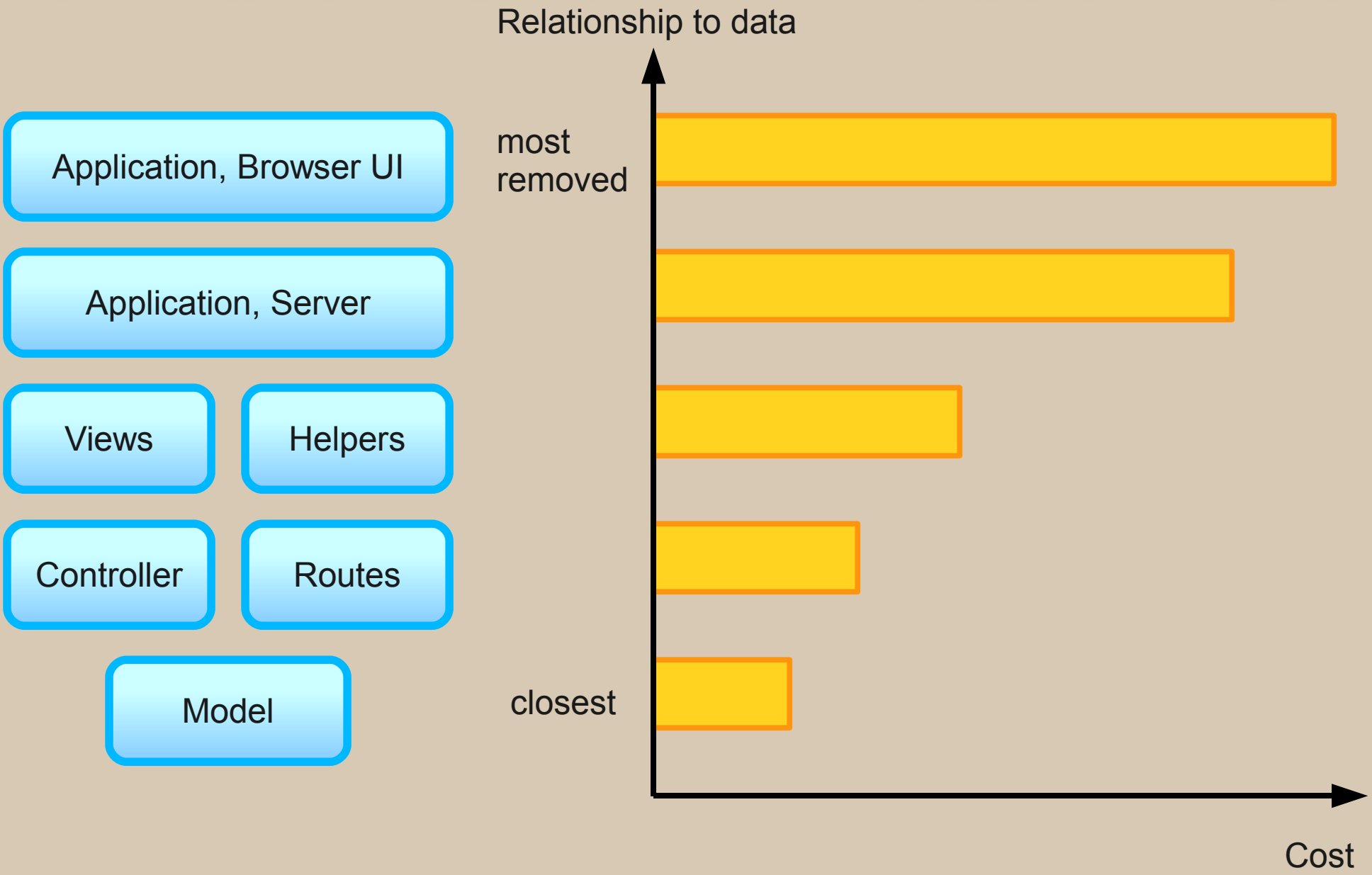


Cost of Testing

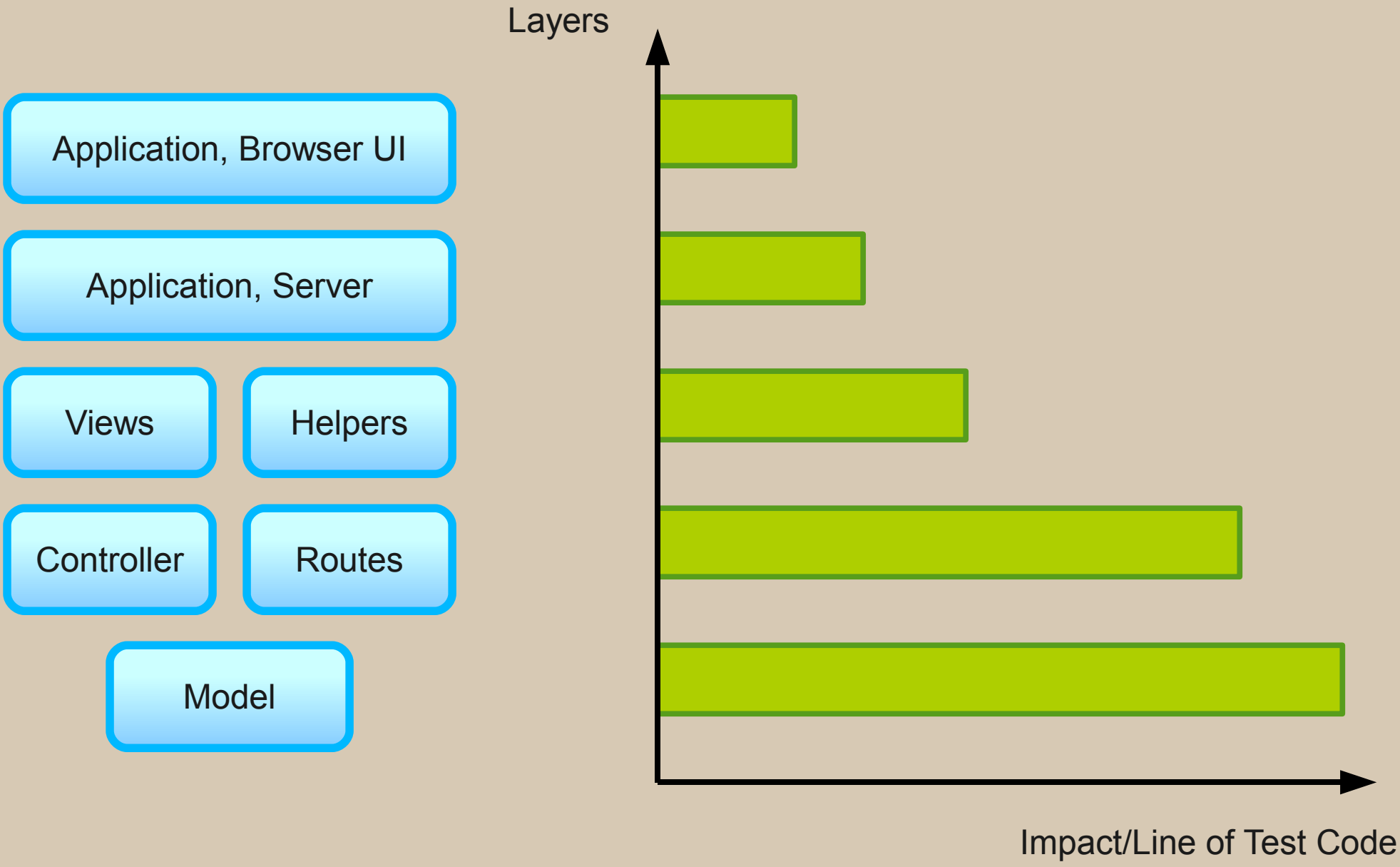
- Application, Browser UI
- Application, Server
- Views
- Helpers
- Controller
- Routes
- Model



Cost of Testing



Best ROI for Testing



TDD & Design Patterns

RubyFocus

Skinny Controller—
Fat Model

DRY

named_scope

Proxy Associations

Validations

...

- Designed to move logic from higher to lower application layers
- Following design patterns makes testing easier
- Code written following TDD economics will naturally converge on these design patterns!

Structure of Tests

Setup

Expected value

Actual value

Verification: actual == expected?

Teardown

Good Tests are...

RubyFocus

Compact

Responsible for testing one concern only

Fast

DRY

RSpec Verifications

RubyFocus

should respond_to

should be_nil

→ works with any ? method (so-called “predicates”)

should be_valid

should_not be_nil; should_not be_valid

lambda {...}.should change(), {}, .from().to(), .by()

should eql, ==, equal

RSpec Structure

RubyFocus

before, before(:each), before(:all)

after, after(:each), after(:all)

describe do...end, nested

it do... end

Let's do some coding

RubyFocus

Demo

Models: What to test?

RubyFocus

Validation Rules

Associations

Any custom method

Association Proxy Methods

Let's do some coding

RubyFocus

Exercise, but wait...

Test-First Teaching

RubyFocus

Pioneered by Sarah Allen

- Instructor writes tests
- Students write code

What about writing tests?

Behavior-First Teaching

RubyFocus

- Instructor specifies desired behavior
  - in plain English
  - like a user story
- Students write tests
- Students write code to make tests pass

Story Exercise #1

A Person object must have a first and last name.

A Person object can construct a full name from the first and last name.

A Person object has an optional middle name.

A Person object returns a full name including, if present, the middle name.

RSpec ==, eql, equal

RubyFocus

obj.should == 5	5 == 5
obj.should eql(5)	5.eql 5
obj.should equal(5)	5.equal 5

Object Equality vs. Identity

eql, ==	compare values
equal, ===	compare objects, classes

Use ==

Unless you know you need something else

Warning! Do not use ***!=*** with RSpec.
Use ***should_not*** instead.

RSpec should change

RubyFocus

```
lambda {
   Person.create
}.should change(Person, :count).by(1)
```

```
lambda {
   @bob.addresses.create(:street => “...”)
}.should change(@bob.addresses, :count).by(1)
```

Warning: The 2nd example is incorrect!

RSpec should change

```
lambda {
   # code that causes a change
}.should change(object, :method).by(difference)
```

or

```
should change(object,
   :method).from(initial).to(final)
```

The object typically is a class, if not, be wary of reload!

should change with block


```
lambda {
   # code that causes a change
}.should change { expression }.by(difference)
```

or

```
should change { expression }.from(initial).to(final)
```

better with block

```
lambda {
   @bob.addresses.create(:street => “...”)
}.should change {
   @bob.addresses(true).count }.by(1)
```

Homework

RubyFocus

A Person should save correctly.

A Person can have many Addresses.

A Person's Address must have a street, city and zip.

A Person's Address can have an optional country.

If the country is left blank, it should default to “USA” prior to saving.

BizConf

Rails & Web App Professionals, Entrepreneurs, Consultants

Small group

Network with Who's Who

Organized by Obie Fernandez of Hashrocket

Highlight: David Allen

Aug 4-6, 2010

Amelia Island, FL

Discount code: WOLF

for 43% off

http://bizconf.org?coupon=WOLF