

Presentation at  
<http://bit.ly/codemash-testing>



Attribution-NonCommercial-ShareAlike 2.0



# Are You Satisfied with Your Tests?

-- or --

Why don't we do it like this ...

Jim Weirich  
Chief Scientist / EdgeCase  
jim@edgecase.com  
@jimweirich





# Testing



**Your Doing it  
All Wrong!!!!**



Survey ...



Java VS .Net VS  
Python VS Ruby  
Developers?

Unit Tests?  
Functional?  
Javascript?  
End to end?

Testing:  
TDD/BDD?  
Unit Testing?  
Any Testing?



Java VS .Net VS  
Python VS Ruby  
Developers?

Unit Tests?  
Functional?  
Javascript?  
End to end?

Are you happy  
with your testing?

Testing:  
TDD/BDD?  
Unit Testing?  
Any Testing?



# ✗ Slow Tests





# Jeff Nielsen

## Psychology of Build Times

- Unit Tests
- Checkin Tests



# Jeff Nielsen

## Psychology of Build Times

- Unit Tests <10 seconds
- Checkin Tests



# Jeff Nielsen

## Psychology of Build Times

- Unit Tests <10 seconds
- Checkin Tests <10 Minutes



# ✗ Blasé Attitude toward Failing Tests





< prev 3461 next > latest >>

3461 (27 Apr)

3459 (27 Apr)

3454 (27 Apr)

3447 (27 Apr) FAILED

3439.1 (27 Apr)

3439 (27 Apr) FAILED

3429 (27 Apr)

3426 (27 Apr)

3416 (27 Apr)

3411 (27 Apr)

3404 (26 Apr) FAILED

3391 (26 Apr) FAILED

3374.1 (26 Apr) FAILED

3374 (26 Apr) FAILED

3350 (23 Apr) FAILED

3340.1 (23 Apr) FAILED

3340 (22 Apr) FAILED

3339 (22 Apr) FAILED

3328 (22 Apr) FAILED

3325 (22 Apr) FAILED

## master build 3461

finished at 9:09 PM on 27 Apr 2010 taking 6 minutes and 58 seconds

### Build Changeset

New revision 3461 detected

Revision 3461 committed by reaton on 2010-04-28 10:32:53

Improved phone forms

M /project/app/views/contracts/\_email.html.haml

M /project/app/views/contracts/phone\_new

Revision 3460 committed by reaton on 2010-04-28 10:32:51

Refactored link generation

### Build Log

### Custom Build Artifacts

cucumber\_coverage

spec\_coverage

test.log

print.css

cucumber.log

requests.log

### Project Settings

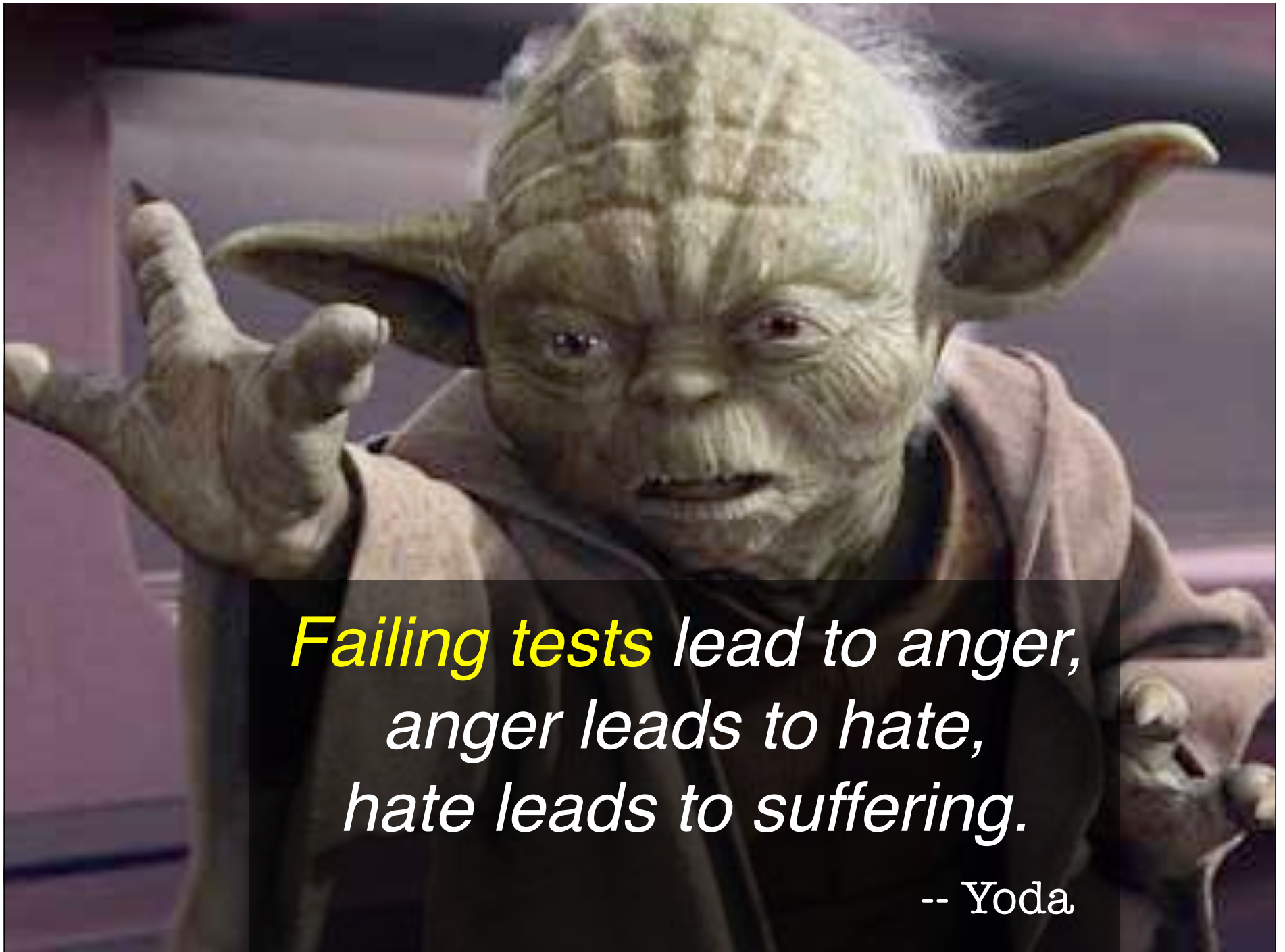




*Fear leads to anger,  
anger leads to hate,  
hate leads to suffering.*

-- Yoda





*Failing tests lead to anger,  
anger leads to hate,  
hate leads to suffering.*

-- Yoda





# TESTS

MAKE SURE THEY'RE ALL GREEN OR ELSE





✗ Over Mocking

Friday, January 14, 2011



# When to Mock

- Using an external service
- Verifying a protocol
- Objects are complicated to create

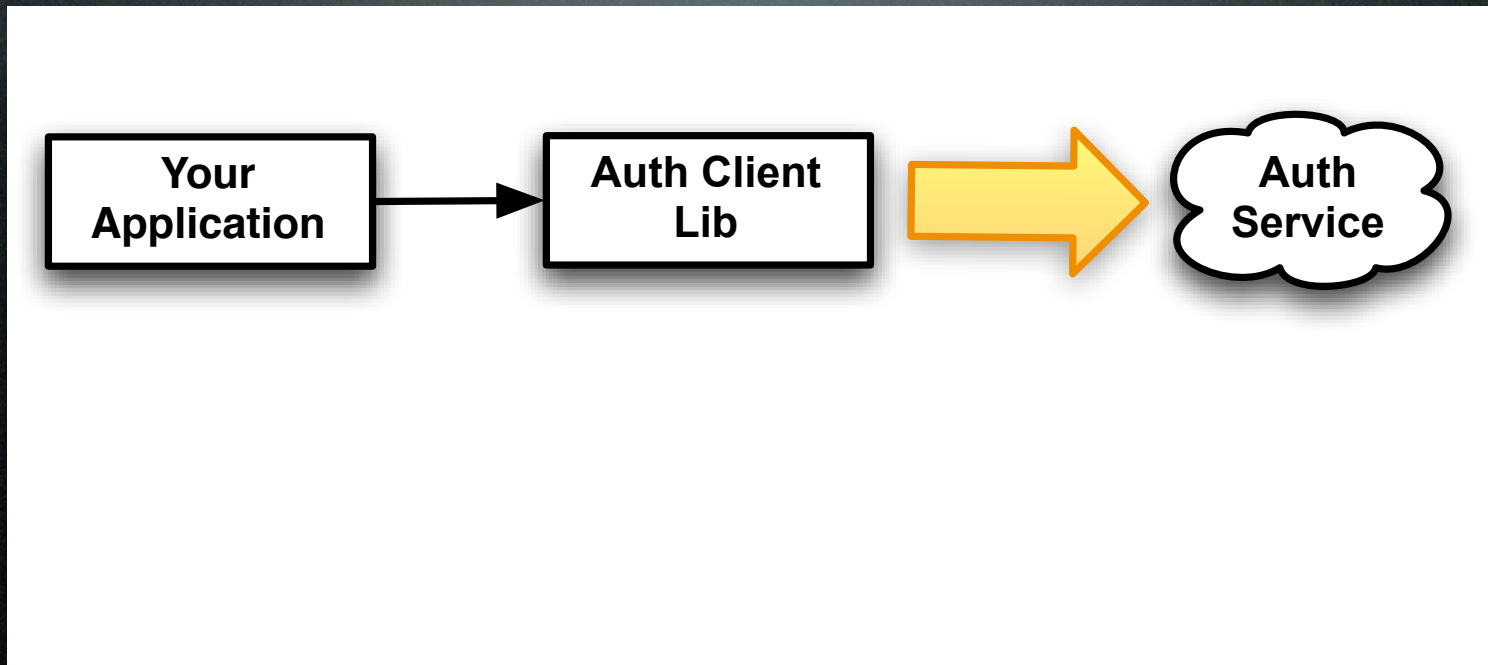


# When to Mock

- Using an external service
- Verifying a protocol
- ~~• Objects are complicated to create~~

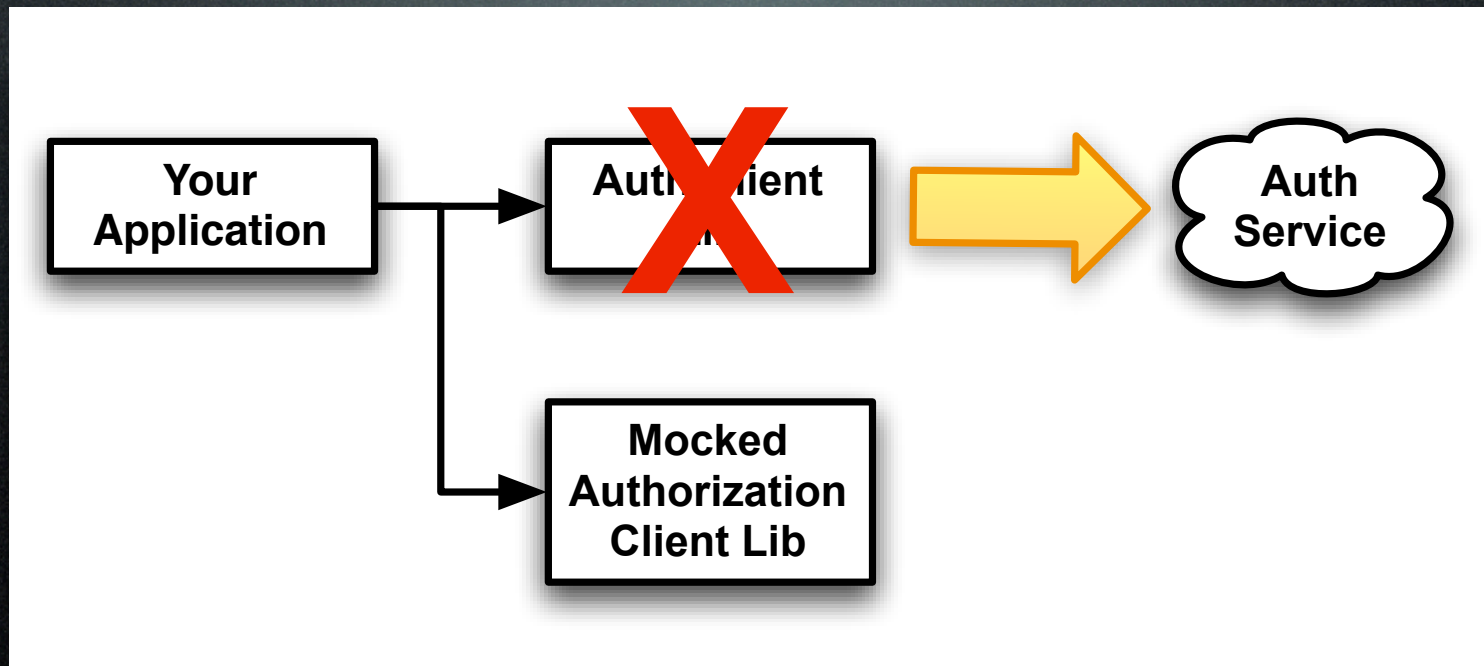


# External Service



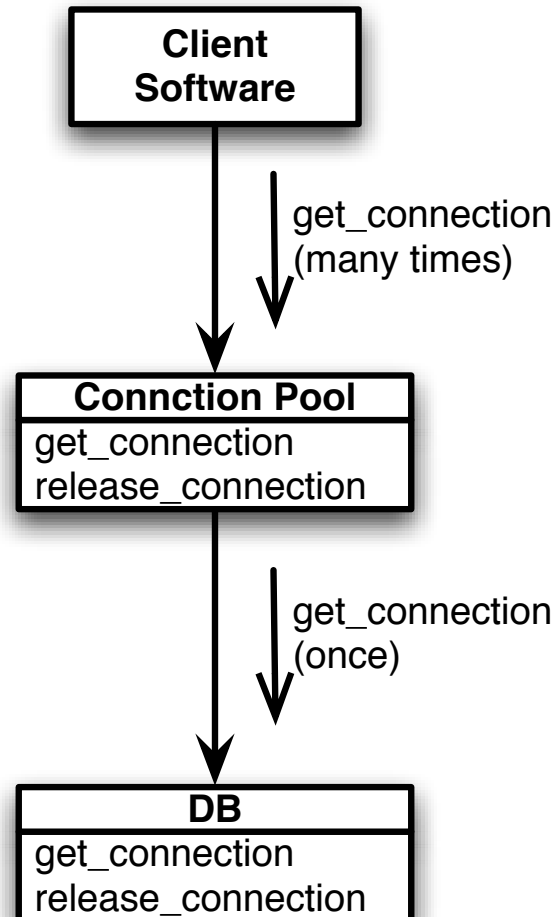


# External Service



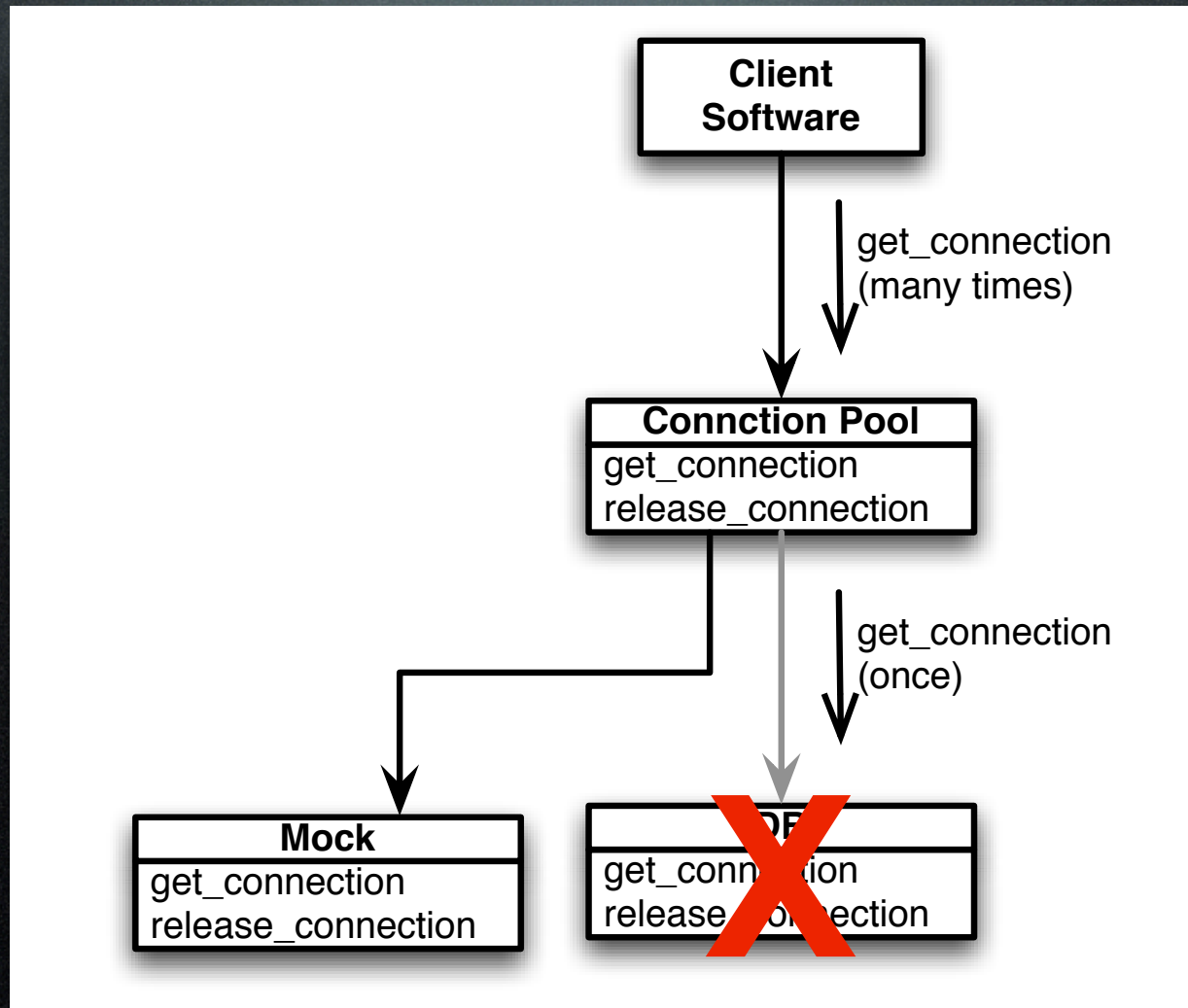


# Verifying a Protocol



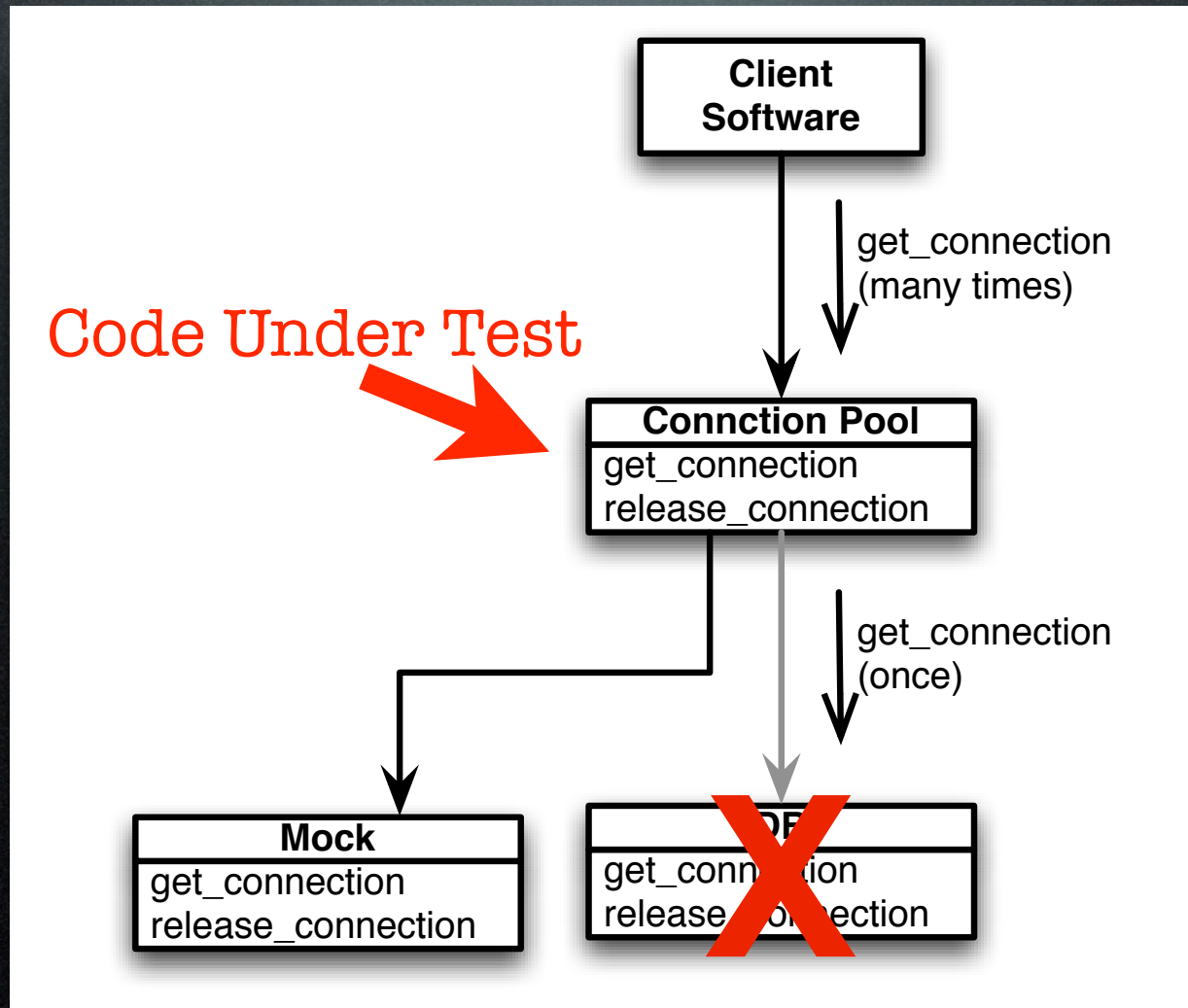


# Verifying a Protocol



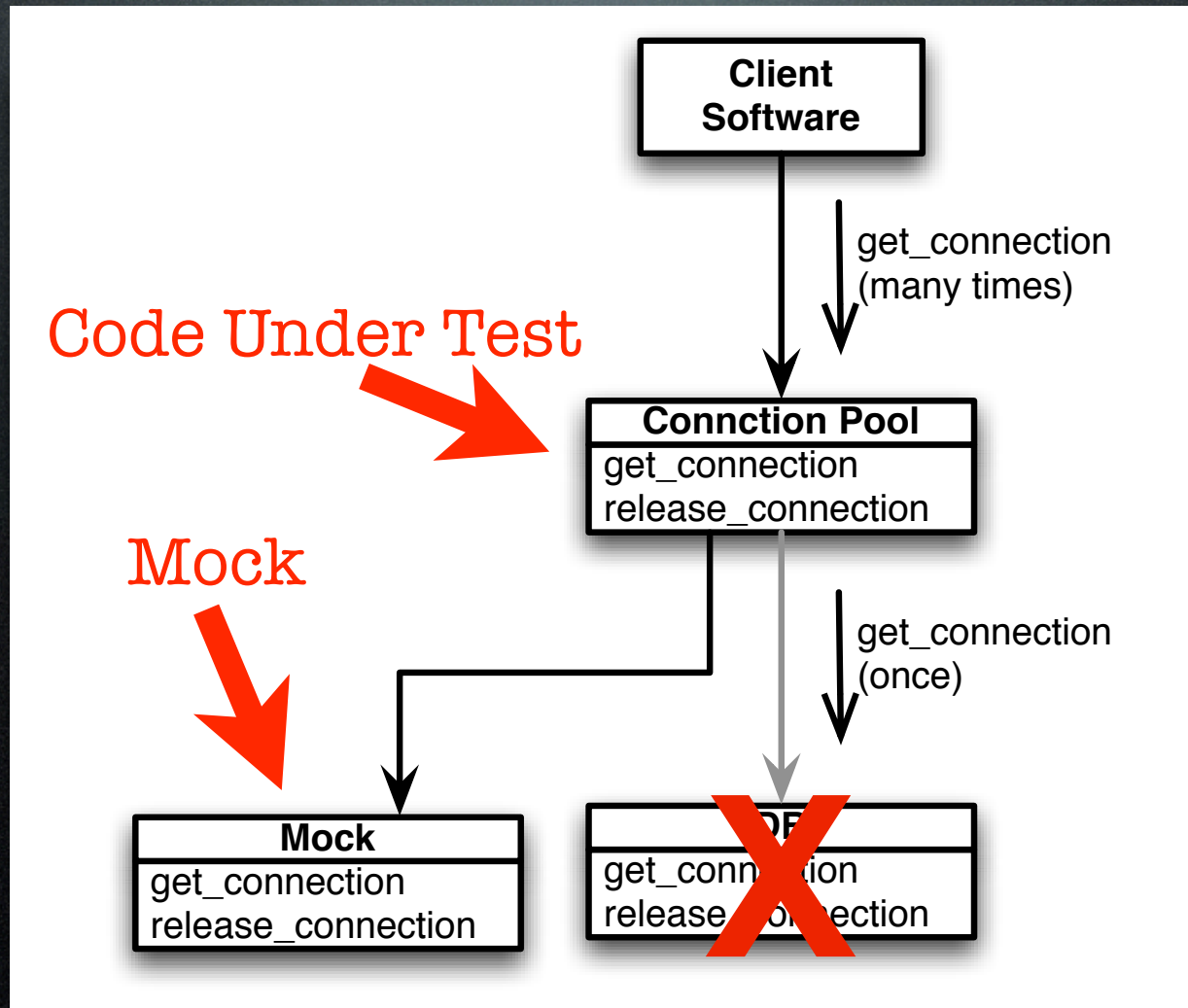


# Verifying a Protocol





# Verifying a Protocol

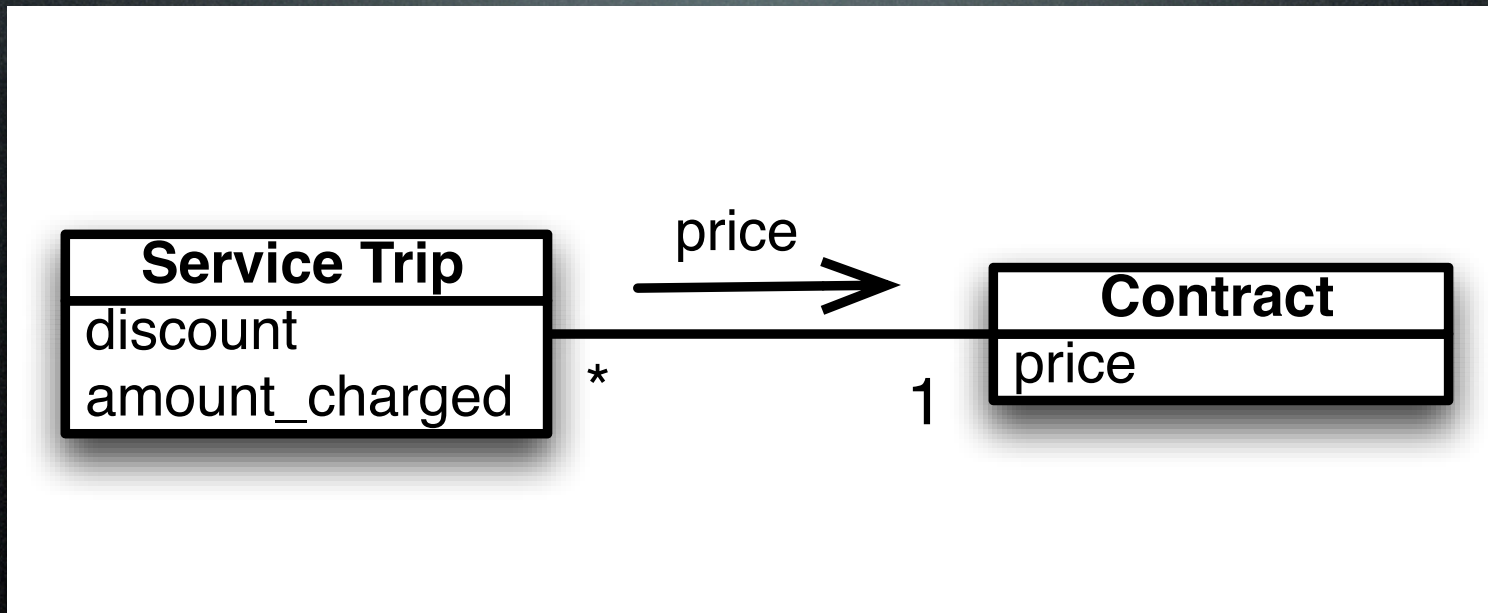




# Overmocking Clues

- You create mocks returning mocks
- You have fantasy tests








```
test "applies discounts to service price" do
  contract = flexmock(:model, Contract,
    :price => 100.0)
  trip = Factory.build(:service_trip,
    :discount => 0.30,
    :contract => contract)

  assert_equal 70.00, trip.amount_charged
end
```



## Mocked to return value

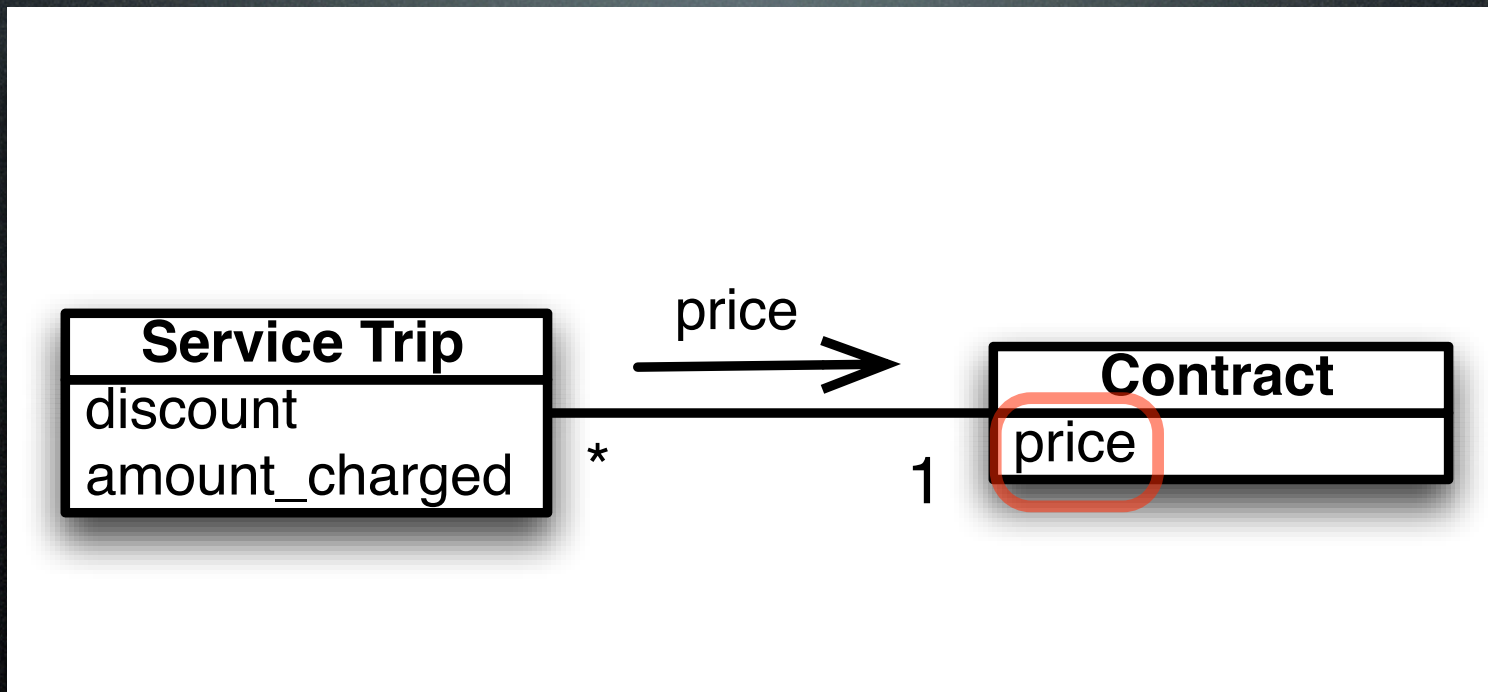


```
test "applies discounts to service price" do
  contract = fleamock(:model, Contract,
    :price => 100.0)
  trip = Factory.build(:service_trip,
    :discount => 0.30,
    :contract => contract)

  assert_equal 70.00, trip.amount_charged
end
```

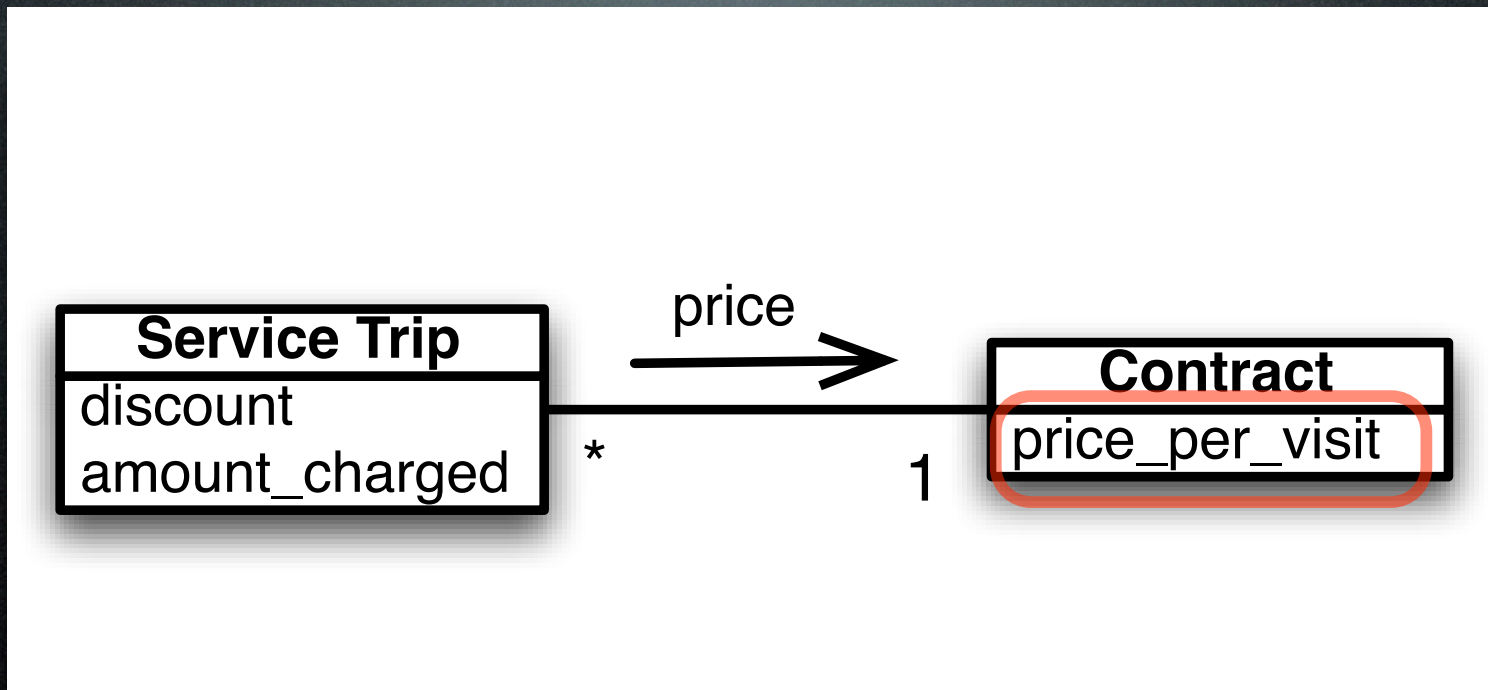


# Refactor!



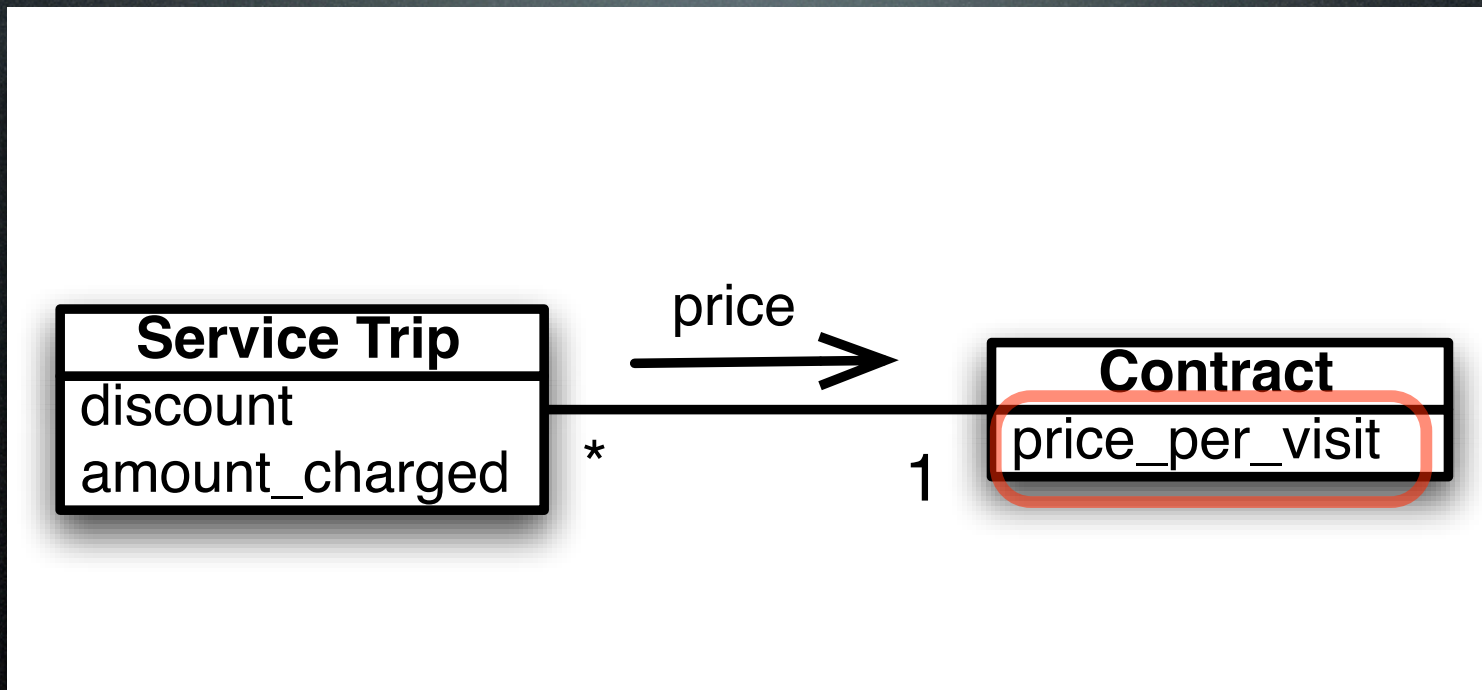


# Refactor!





# Refactor!

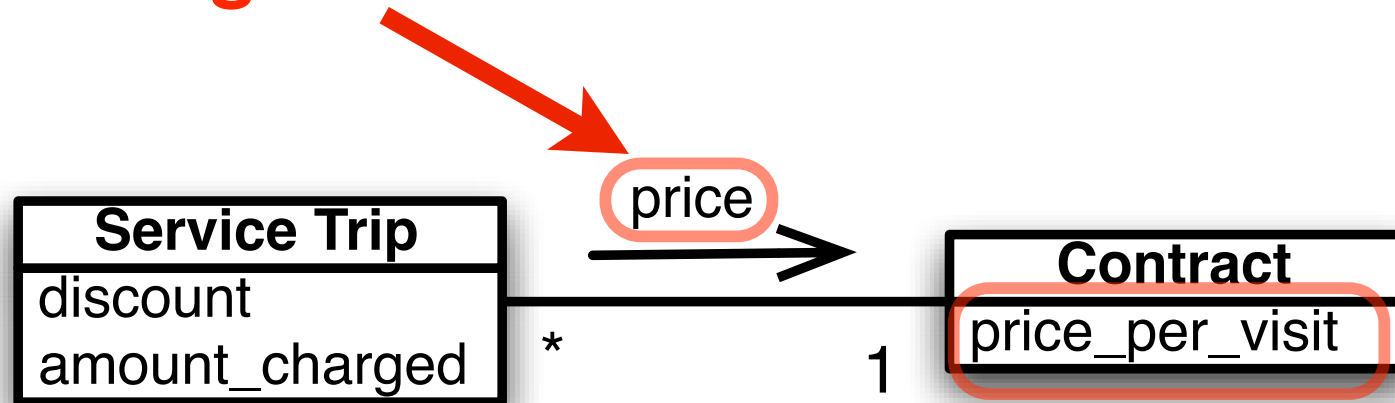


1 tests, 1 assertions, 0 failures, 0 errors, 0 skips



# Refactor!

**Wrong method name!**



1 tests, 1 assertions, 0 failures, 0 errors, 0 skips



# Fix Service Trip

**Correct Method Name**





# Fix Service Trip

**Correct Method Name**



NoMethodError: undefined method `price\_per\_visit'  
for "#<FlexMock>":Service  
1 tests, 0 assertions, 0 failures, 1 errors, 0 skips



# Fantasy Tests

- Pass when the code is incorrect.
- Fail when the code is correct.



# Summary



- Use a mock to
  - Mock an external service
  - Verify a protocol



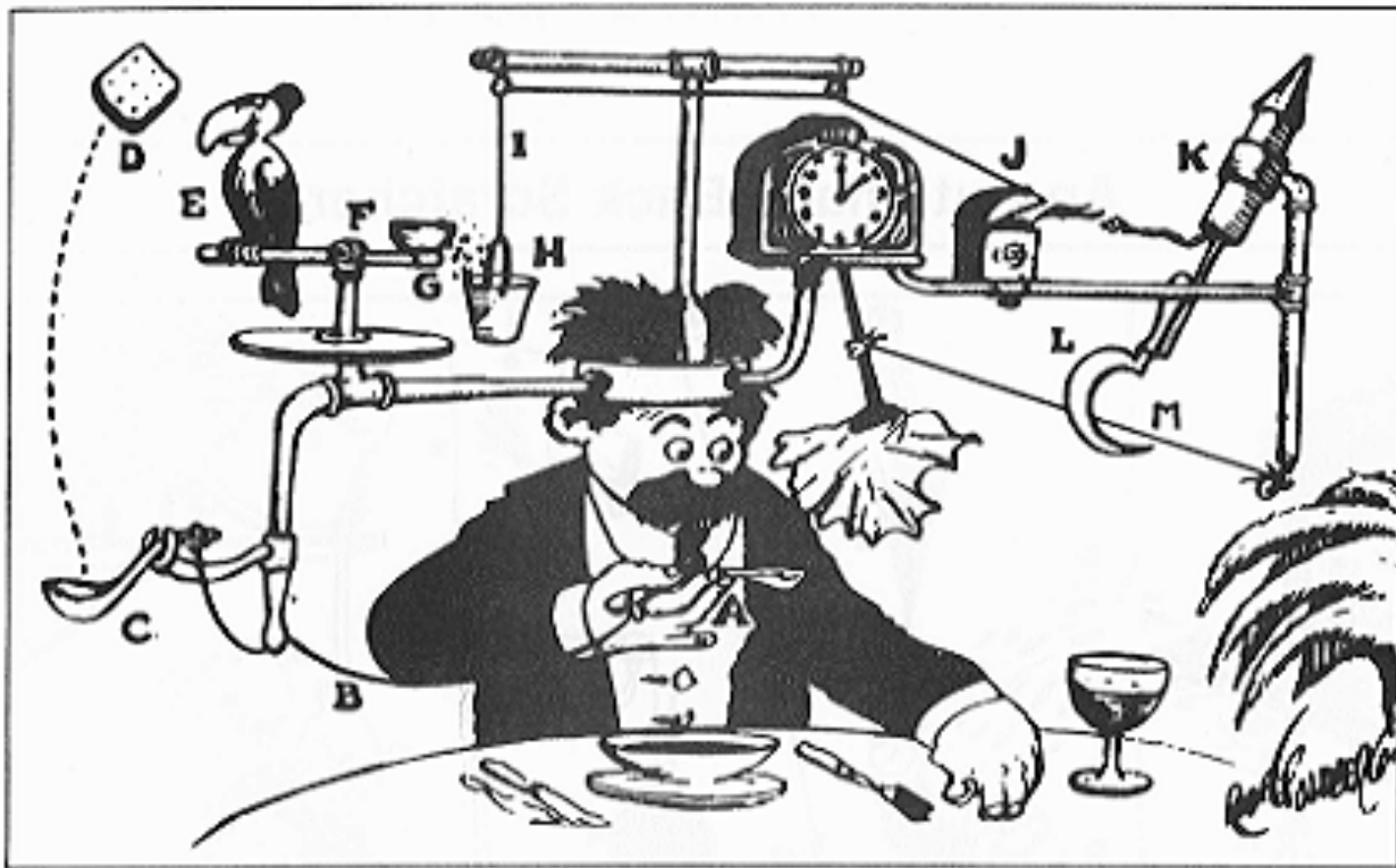
- Don't use a mock to:
  - Avoid creating complex Objects





# Complex Object Builds

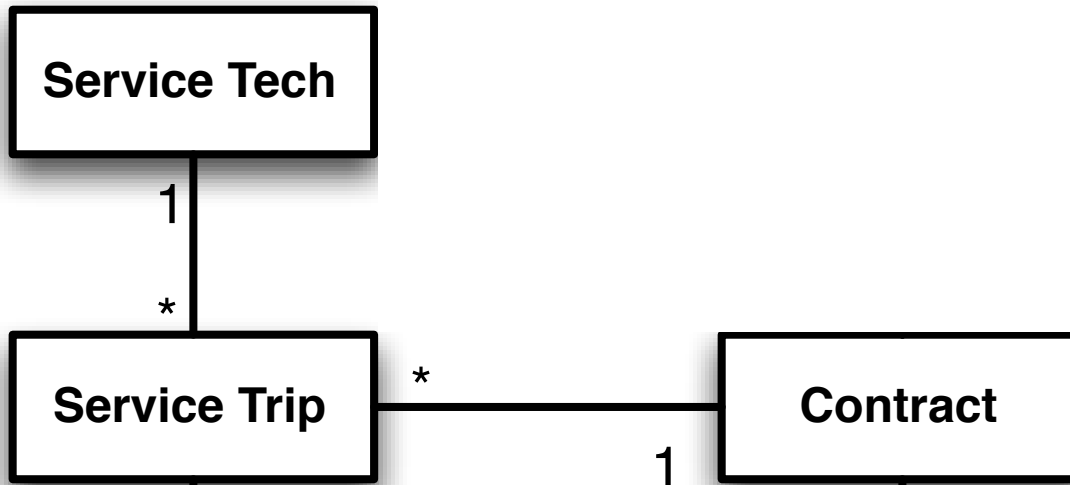
Self-Operating Napkin



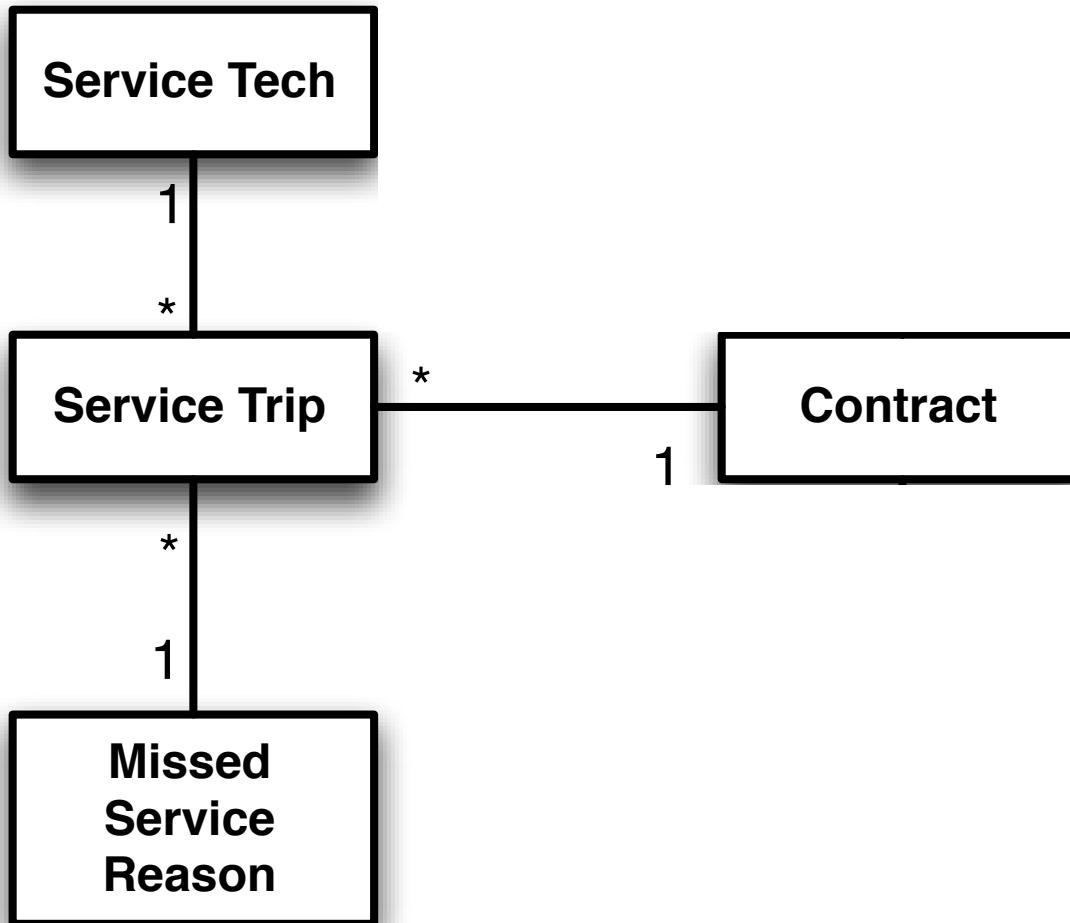




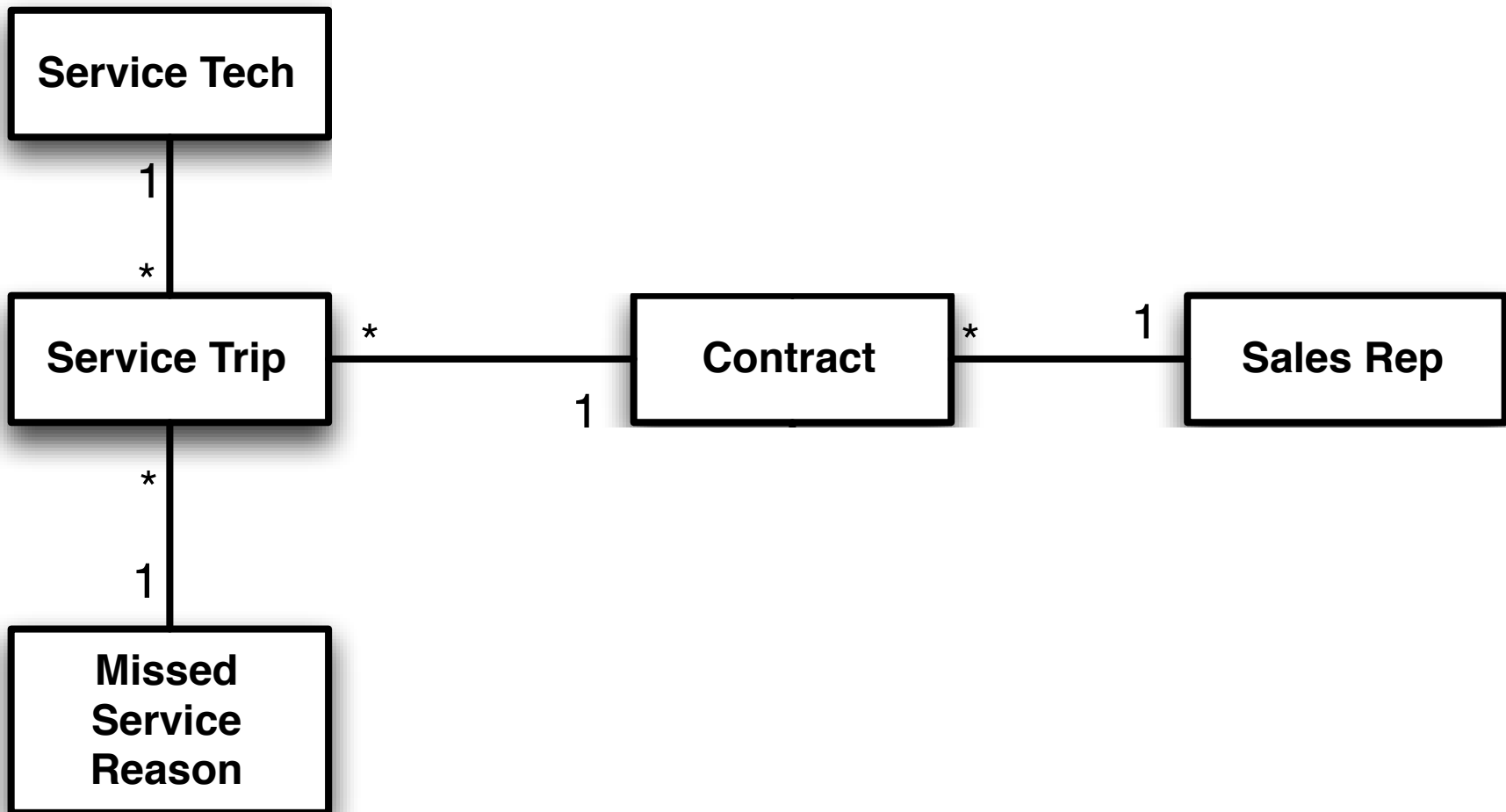




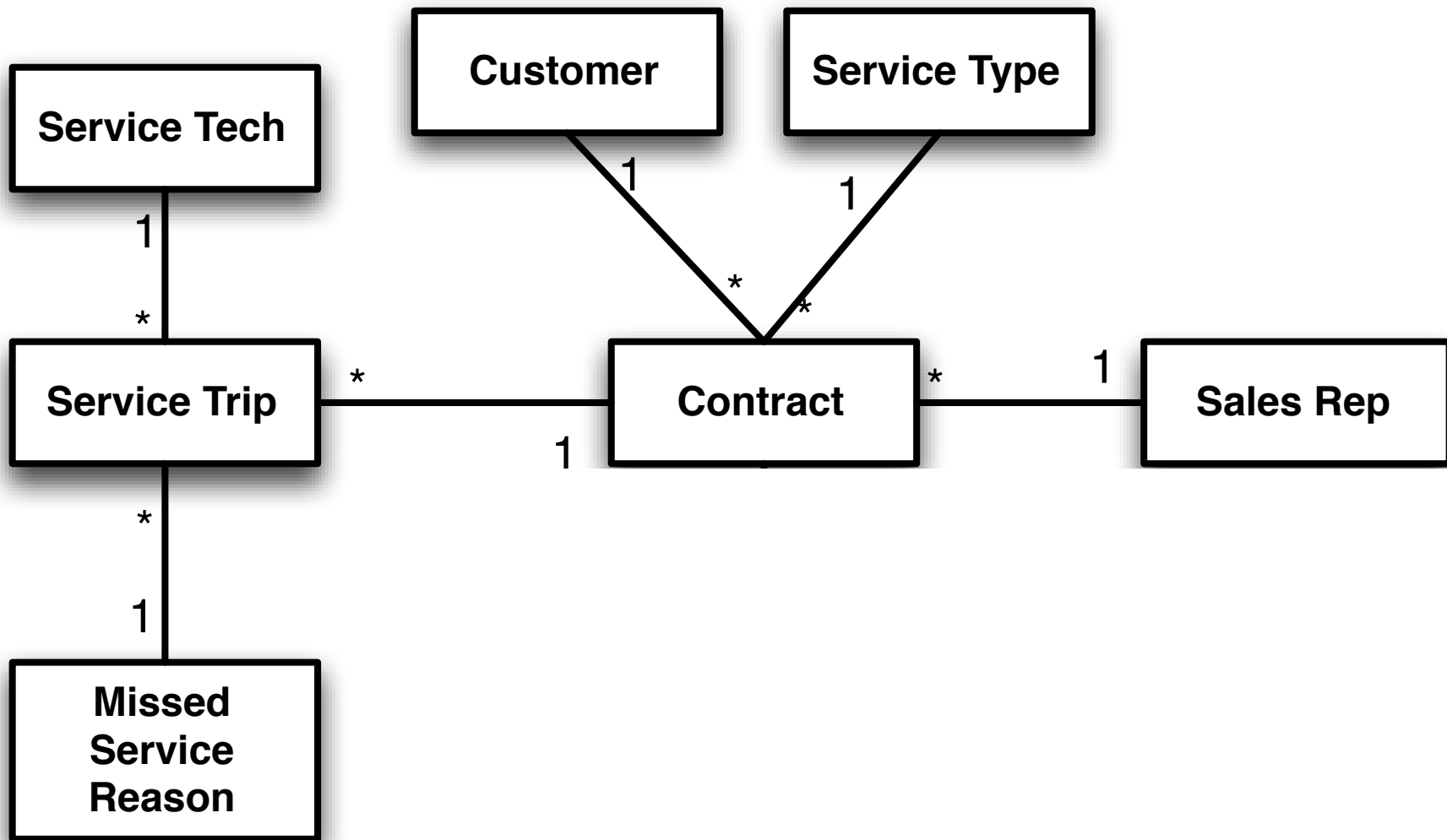




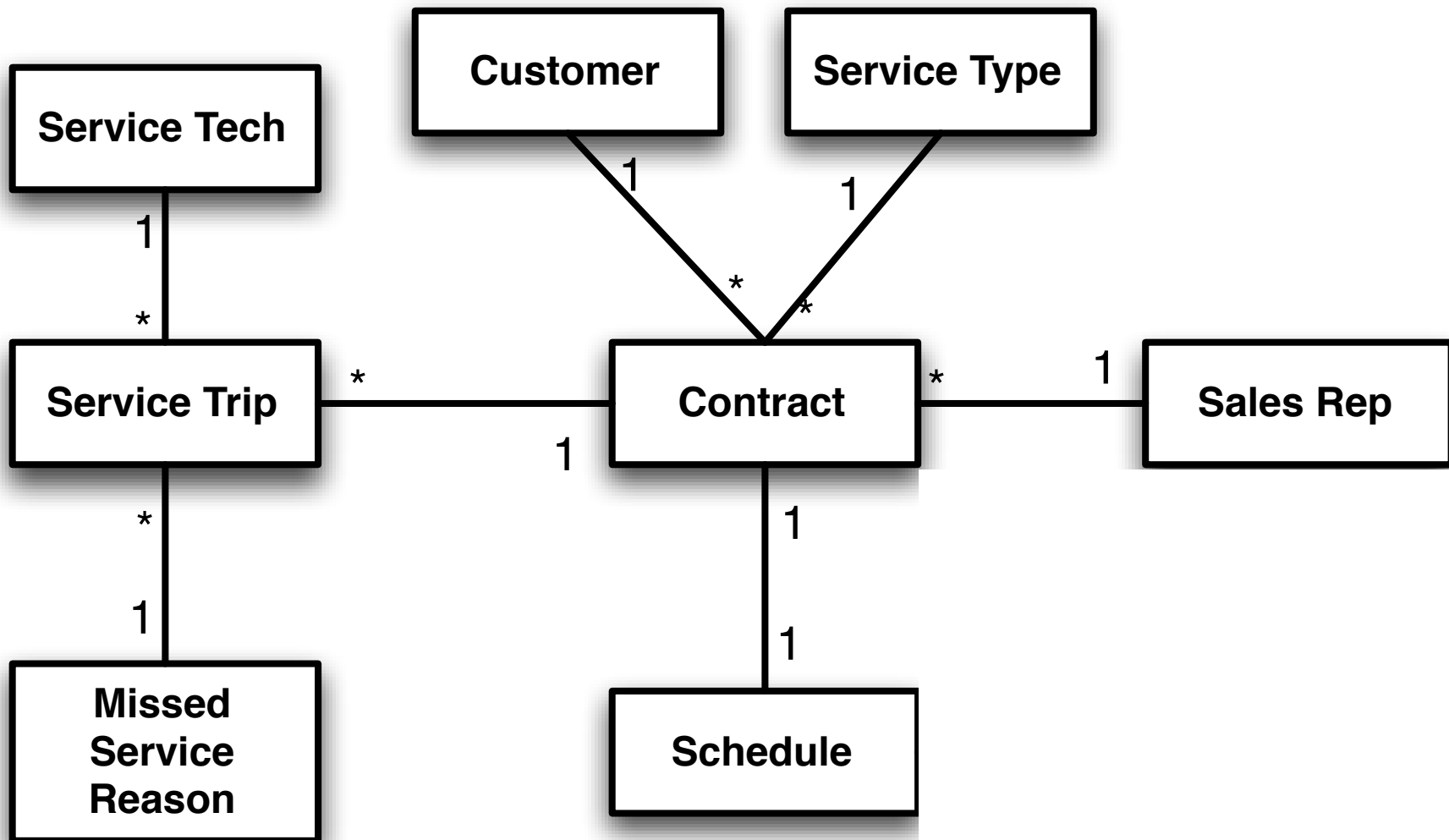




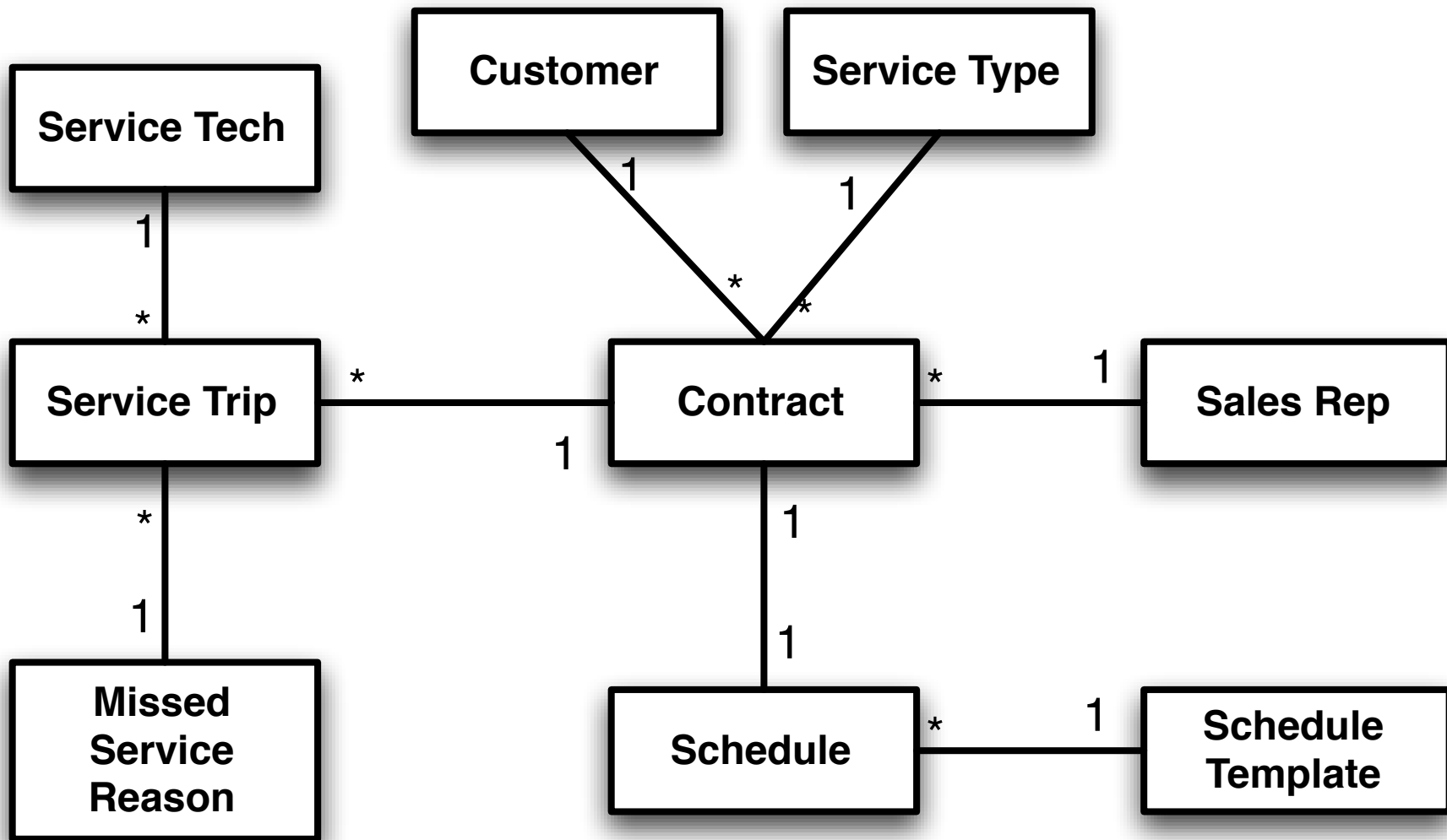














```
Factory.define :service_trip do |trip|  
  trip.association :service_tech  
  trip.association :missed_service_reason  
  trip.association :contract  
  trip.discount 0.0  
end
```



# Create in Database

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.create(:service_trip)
    end
  }
end
```



# Create in Database

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.create(:service_trip)
    end
  }
end
```

Time: 4.75 Seconds



# Faster Database (sqlite3)

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.create(:service_trip)
    end
  }
end
```



# Faster Database (sqlite3)

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.create(:service_trip)
    end
  }
end
```

Time: 4.37 Seconds




# Factory In-Memory

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.build(:service_trip)
    end
  }
end
```



# Factory In-Memory

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.build(:service_trip)
    end
  }
end
```




**Build In Memory**



# Factory In-Memory

```
test "Time for Factory.create" do
  bench("Factory.create") {
    TIMES.times do
      trip = Factory.build(:service_trip)
    end
  }
end
```

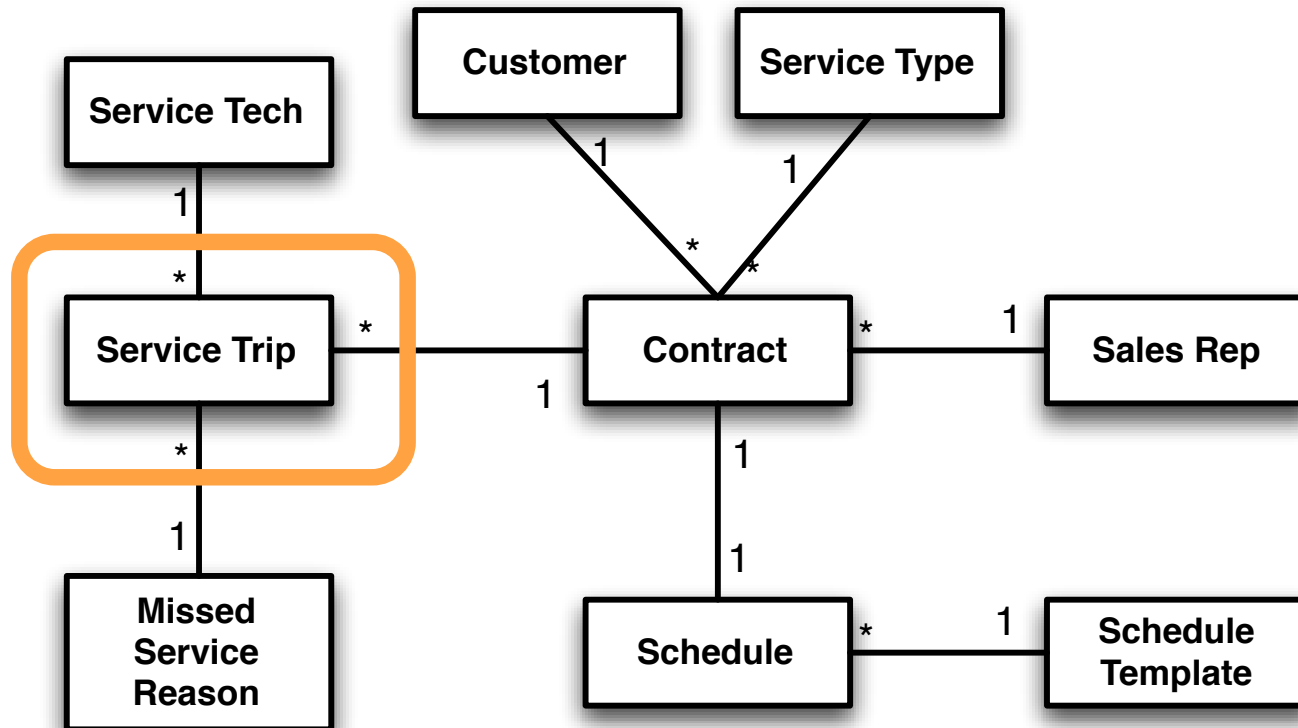


**Build In Memory**

Time: 3.98 Seconds

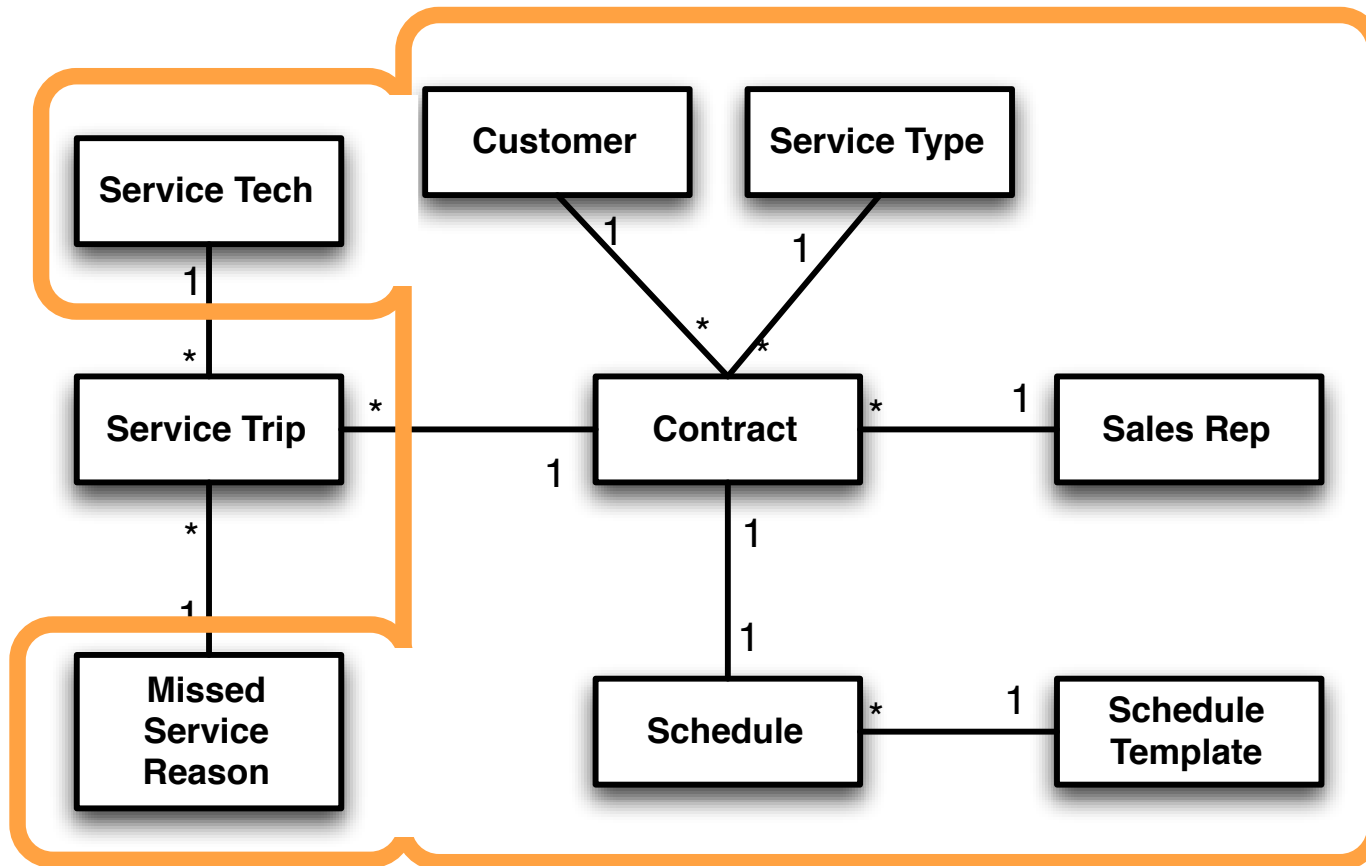


# Built In Memory





## Built In Database





# Using Mocks

```
test "Time for Mocking" do
  bench("Flexmock") {
    TIMES.times do
      trip = Factory.build(:service_trip,
        :missed_service_reason =>
          flexmock(:model, MissedServiceReason),
        :service_tech =>
          flexmock(:model, ServiceTech),
        :contract => flexmock(:model, Contract))
    end
  }
end
```



# Using Mocks

```
test "Time for Mocking" do
  bench("Flexmock") {
    TIMES.times do
      trip = Factory.build(:service_trip,
        :missed_service_reason =>
          flexmock(:model, MissedServiceReason),
        :service_tech =>
          flexmock(:model, ServiceTech),
        :contract => flexmock(:model, Contract))
    end
  }
end
```

Time: 0.59 seconds



# Using Factory.attributes\_for

```
test "Time for Custom Factory" do
  bench("Factory attributes") {
    TIMES.times do
      attrs = Factory.attributes_for(:service_trip)
      attrs.merge(
        :missed_service_reason => ...,
        :service_tech => ...,
        :contract => ...)
      trip = ServiceTrip.new(attrs)
    end
  }
end
```



# Using Factory.attributes\_for

```
test "Time for Custom Factory" do
  bench("Factory attributes") {
    TIMES.times do
      attrs = Factory.attributes_for(:service_trip)
      attrs.merge(
        :missed_service_reason => ...,
        :service_tech => ...,
        :contract => ...)
      trip = ServiceTrip.new(attrs)
    end
  }
end
```

Time: 0.30 seconds



# Custom In-Memory

```
test "Time for Custom Build" do
  bench("Custom") {
    TIMES.times do
      trip = ServiceTrip.new(
        :missed_service_reason =>
          MissedServiceReason.new,
        :service_tech => ServiceTech.new,
        :contract => Contract.new)
    end
  }
end
```



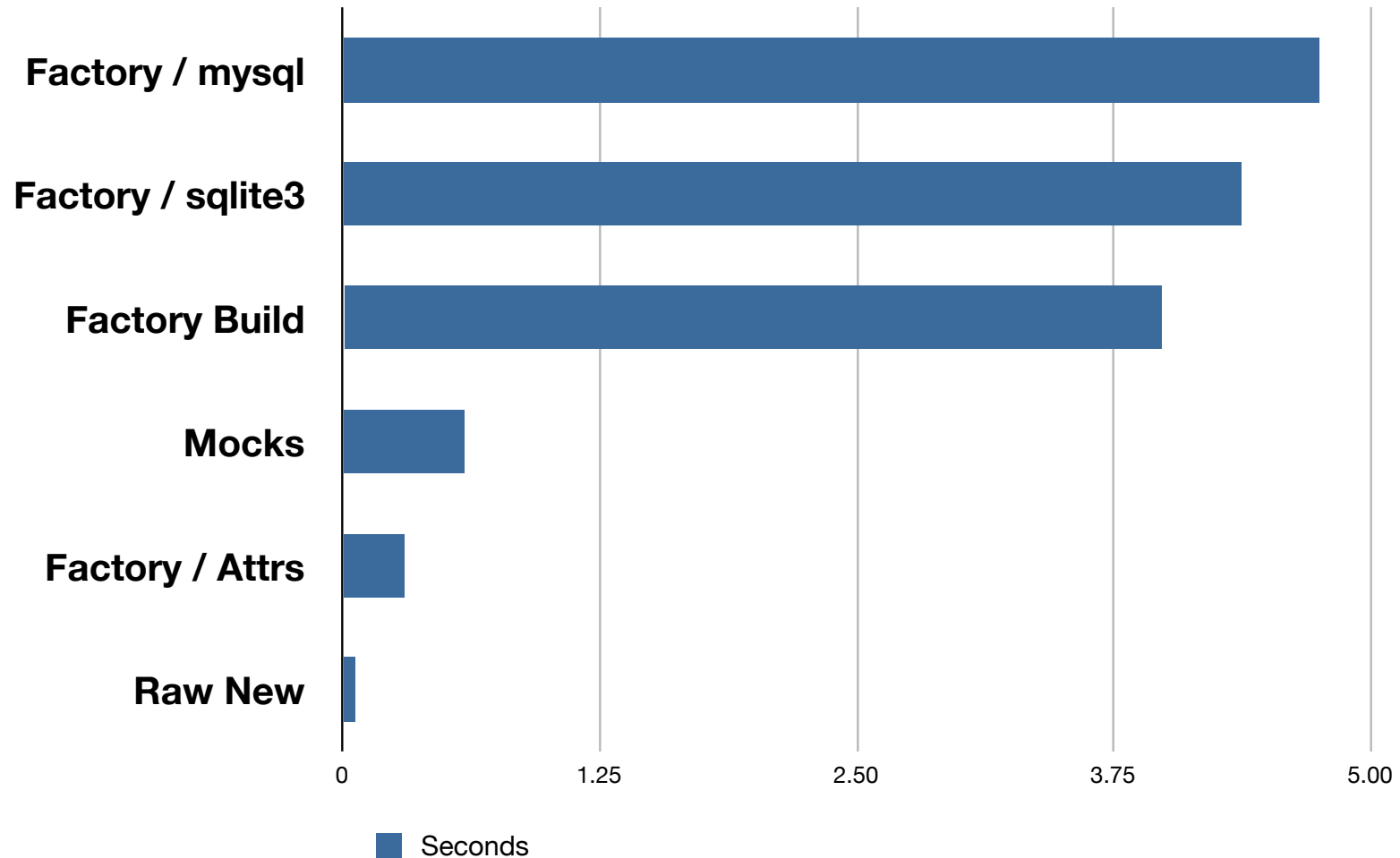
# Custom In-Memory

```
test "Time for Custom Build" do
  bench("Custom") {
    TIMES.times do
      trip = ServiceTrip.new(
        :missed_service_reason =>
          MissedServiceReason.new,
        :service_tech => ServiceTech.new,
        :contract => Contract.new)
    end
  }
end
```

Custom: 0.06 Seconds



# Timing Summary





# ✗ Gratuitous Use of the Database

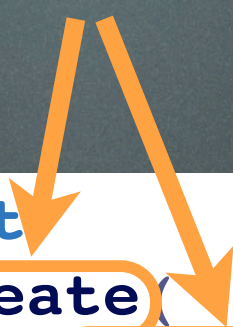




```
def test_total_cost
  order = Order.create(
    :items => [Item.create(:cost => 10)])
  assert_equal 10, order.total_cost
end
```



## In the Database?



```
def test_total_cost
  order = Order.create(
    :items => [Item.create(:cost => 10)])
  assert_equal 10, order.total_cost
end
```



```
def test_total_cost
  order = Order.new(
    :items => [Item.new(:cost => 10)])
  assert_equal 10, order.total_cost
end
```



# save VS valid?

```
def test_order_fails_with_bad_supplier
  order = Order.new(:supplier => :bad)
  assert ! order.save
end
```



# save VS valid?

```
def test_order_fails_with_bad_supplier
  order = Order.new(:supplier => :bad)
  assert ! order.valid?
end
```



# save VS valid?

```
def test_order_fails_with_bad_supplier
  order = Order.new(:supplier => :bad)

  assert ! order.valid?
  assert model.errors.on(:supplier)
  assert_match(/(invalid|bad).*supplier/i,
    model.errors.on(field).to_s,
end
```





# Custom Assertions





```
def assert_tween(min, max, actual, name)
  assert actual >= min,
    "#{name} must be >= #{min} (was #{actual})"
  assert actual <= max,
    "#{name} must be <= #{max} (was #{actual})"
end
```



```
should 'be randomly distributed' do
  collect_face_counts.each do |face, count|
    assert_tween 1, 6, face, "face"
    assert_tween 800, 1200, count, "count"
  end
end
```



```
def assert_validation_error_on(model,
                               field=nil,
                               pattern=//)

  if field
    assert ! model.valid?
    assert model.errors.on(field)
    assert_match(re,
                 model.errors.on(field).to_s)
  else
    assert ! model.valid?
    assert_match(re,
                 model.errors.full_messages.join(", "))
  end
end
```

(note: real version has custom error messages)



# ✗ Over Meta in Tests



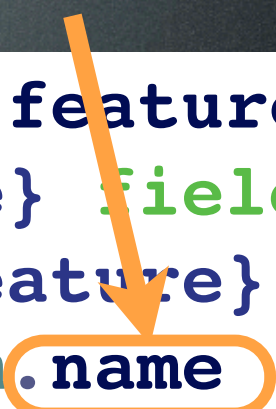


```
%w(name address).each do |feature|  
  it "clears the #{feature} field" do  
    @item.send("clear_#{feature}")  
    assert_equal "", @item.name  
  end  
end
```



## Explicit Reference

```
%w(name address).each do |feature|  
  it "clears the #{feature} field" do  
    @item.send("clear_#{feature}")  
    assert_equal "", @item.name  
  end  
end
```





```
it "clears the name field" do
  @item.clear_name
  assert_equal "", @item.name
end

it "clears the address field" do
  @item.clear_address
  assert_equal "", @item.address
end
```



# ✖ Testing Private Methods





```
describe :load_personal_data do
  before do
    @entry = stubbed_entry
    @entry.stub!(:owner).and_return(:owner_id)
    controller.instance_variable_set('@entry', @entry)
  end

  it "loads the personal data from from the" do
    controller.stub!(:params).
      and_return(:person => 'John Doe')
    PersonalDataService.
      should_receive(:get_personal_data).
      with(:owner_id)

    controller.send(:load_personal_data)
  end
end
```



```
describe :load_personal_data do
  before do
    @entry = stubbed_entry
    @entry.stub!(:owner).and_return(:owner_id)
    controller.instance_variable_set('@entry', @entry)
  end
```



## Pathological Coupling

```
it "loads the personal data" do
  controller.stub!(:params).
    and_return(:person => 'John Doe')
  PersonalDataService.
    should_receive(:get_personal_data).
    with(:owner_id)

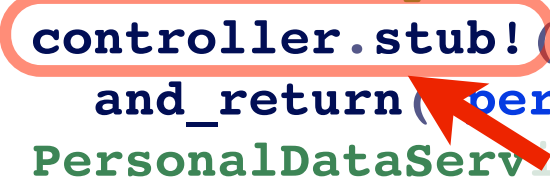
  controller.send(:load_personal_data)
end
end
```



```
describe :load_personal_data do
  before do
    @entry = stubbed_entry
    @entry.stub!(:owner).and_return(:owner_id)
    controller.instance_variable_set('@entry', @entry)
  end

  it "loads the personal data from from the" do
    controller.stub!(:params).
      and_return(person => 'John Doe')
    PersonalDataService
      should_receive(:get_personal_data).
        with(:owner_id)

    controller.send(:load_personal_data)
  end
end
```




**More Pathological Coupling**



```
describe :load_personal_data do
  before do
    @entry = stubbed_entry
    @entry.stub!(:owner).and_return(:owner_id)
    controller.instance_variable_set('@entry', @entry)
  end

  it "loads the personal data from from the" do
    controller.stub!(:params).
      and_return(:person => 'John Doe')
    PersonalDataService.
      should_receive(:get_personal_data)
      with(:owner_id)
      controller.send(:load_personal_data)
  end
end
```

**Bypass Normal Privacy Controls**





# In Campfire ...

**Jim W:** Move it to another class and test that class  
Testing private controller methods via send  
makes controller tests WAY too brittle.

**Scott B:** it also kills unicorns  
so – congratulations. now they're extinct.















# ✗ Incorrect use of Describe / Context





```
it "clears the name field" do
  @item.clear_name
  assert_equal "", @item.name
end
```

```
it "clears the address field" do
  @item.clear_address
  assert_equal "", @item.address
end
```



```
describe Item do
  describe "#clear_name" do
    it "clears the name field" do
      ...
    end
  end
end
describe "#clear_address" do
  it "clears the address field" do
    ...
  end
end
end
```



```
describe "#score" do
  before { @bowling = BowlingScorer.new }
  context "with no throws" do
    before { @throws = [] }
    it "returns zero" do
      @bowling.score(@throws).should == 0
    end
  end
  context "with one throw" do
    before { @throws = [9] }
    it "returns the throw" do
      @bowling.score(@throws).should == 9
    end
  end
end
```



```
describe "#score" do
  before { @bowling = BowlingScorer.new }
  context "with no throws" do
    before { @throws = [] }
    it "returns zero" do
      @bowling.score(@throws).should == 0
    end
  end
  context "with one throw" do
    before { @throws = [9] }
    it "returns the throw" do
      @bowling.score(@throws).should == 9
    end
  end
end
```



```
describe "#score" do
  before { @bowling = BowlingScorer.new }
  context "with no throws" do
    before { @throws = [] }
    it "returns zero" do
      @bowling.score(@throws).should == 0
    end
  end
  context "with one throw" do
    before { @throws = [9] }
    it "returns the throw" do
      @bowling.score(@throws).should == 9
    end
  end
end
```



```
describe "#score" do
  before { @bowling = BowlingScorer.new }
  context "with no throws" do
    before { @throws = [] }
    it "returns zero" do
      @bowling.score(@throws).should == 0
    end
  end
  context "with one throw" do
    before { @throws = [9] }
    it "returns the throw" do
      @bowling.score(@throws).should == 9
    end
  end
end
```



# Guidelines

- Use **describe** with ...
  - Things
  - (class names, method names)
- Use **context** with ...
  - Situations
  - (when ..., with ...)





# Refactoring Tests



Friday, January 14, 2011




```
describe "#score" do
  before { @bowling = BowlingScorer.new }
  context "with no throws" do
    before { @throws = [] }
    it "returns zero" do
      @bowling.score(@throws).should == 0
    end
  end
  context "with one throw" do
    before { @throws = [9] }
    it "returns the throw" do
      @bowling.score(@throws).should == 9
    end
  end
end
```

```
describe "#score" do
  let(:bowling) { BowlingScorer.new }
  context "with no throws" do
    let(:throws) { [] }
    it "returns zero" do
      bowling.score(throws).should == 0
    end
  end
  context "with one throw" do
    let(:throws) { [9] }
    it "returns the throw" do
      bowling.score(throws).should == 9
    end
  end
end
```



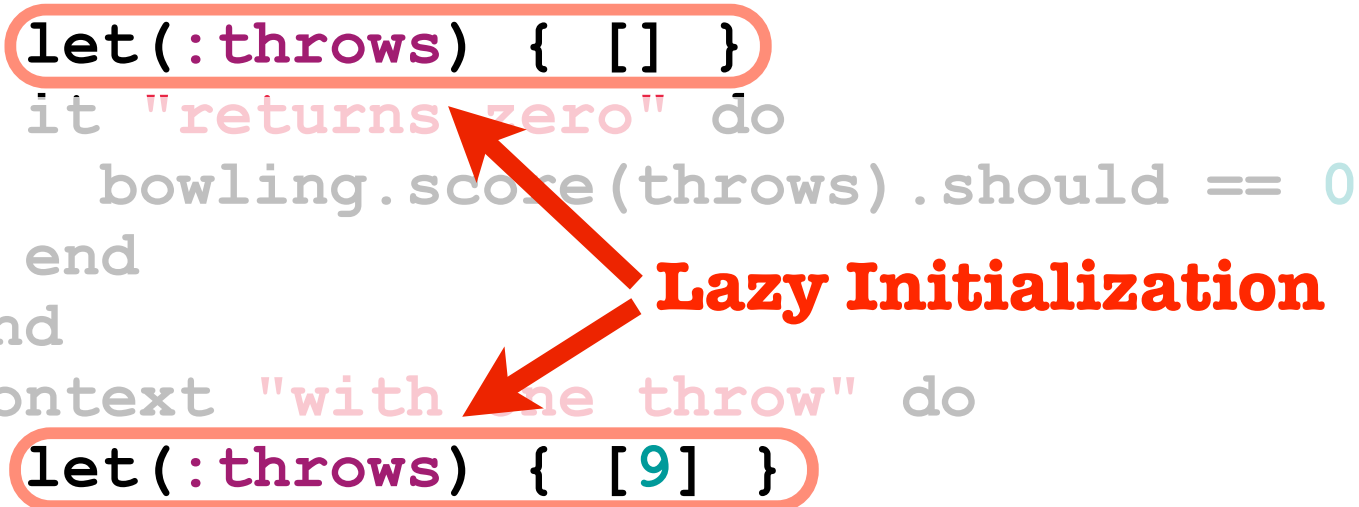
```
describe "#score" do
  let(:bowling) { BowlingScorer.new }
  context "with no throws" do
    let(:throws) { [] }
    it "returns zero" do
      bowling.score(throws).should == 0
    end
  end
end
context "with one throw" do
  let(:throws) { [9] }
  it "returns the throw" do
    bowling.score(throws).should == 9
  end
end
end
```



## Lazy Initialization

```
describe "#score" do
  let(:bowling) { BowlingScorer.new }
  context "with no throws" do
    let(:throws) { [] }
    it "returns zero" do
      bowling.score(throws).should == 0
    end
  end
end
context "with one throw" do
  let(:throws) { [9] }
  it "returns the throw" do
    bowling.score(throws).should == 9
  end
end
end
```

**Lazy Initialization**





```
describe "#score" do
  let(:bowling) { BowlingScorer.new }
  context "with no throws" do
    let(:throws) { [] }
    it "returns zero" do
      bowling.score(throws).should == 0
    end
  end
  context "with one throw" do
    let(:throws) { [9] }
    it "returns the throw" do
      bowling.score(throws).should == 9
    end
  end
end
```

**Using Lazy Initializations**

A diagram with the text "Using Lazy Initializations" in bold red font. Four red arrows point from this text to the expressions "bowling.score(throws)" in the two test cases of the code. Each expression is enclosed in a red rounded rectangle. The first arrow points to the "bowling.score(throws)" in the "with no throws" context, and the second arrow points to the "bowling.score(throws)" in the "with one throw" context.



```
it "should return the throw" do  
  bowling.score(throws).should == 9  
end
```



```
it "should return the throw" do
  bowling.score(throws).should == 9
end
```




```
it "returns the throw" do  
  bowling.score(throws).should == 9  
end
```




```
describe Stack do
  context "stack with one item" do
    let(:stack) { a_stack_with_one_item }
    context "when popped" do
      before { stack.pop }
      it "is empty" do
        stack.should be_empty
      end
    end
  end
end
end
```

```
describe Stack do
  context "stack with one item" do
    let(:stack) { a_stack_with_one_item }
    context "when popped" do
      before { stack.pop }
      it "is empty" do
        stack.should be_empty
      end
    end
  end
end
end
```






```
describe Stack do
  context "stack with one item" do
    let(:stack) { a_stack_with_one_item }
    context "when popped" do
      before { stack.pop }
      it "is empty" do
        stack.should be_empty
      end
    end
  end
end
end
```



```
describe Stack do
  context "stack with one item" do
    let(:stack) { a_stack_with_one_item }
    context "when popped" do
      before { stack.pop }
      it "is empty" do
        stack.should be_empty
      end
    end
  end
end
end
```





```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
```

# Declare Subject




```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
```




# Explicit Use

```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
```



```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
end
```



**Implicit Use**



# What Code is Under Test?

```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
```

# What Code is Under Test?

## What Code is Setup?

```
describe Stack do
  context "nearly empty" do
    subject { a_stack_with_one_item }
    context "when popped" do
      before { subject.pop }
      it { should be_empty }
    end
  end
end
```



# gem rspec-given

```
describe Stack do
  context "nearly empty" do
    Given(:stack) { a_stack_with_one_item }
    When { stack.pop }
    Then { stack.should be_empty }
  end
end
```



## EARLY COLUMBIA SERIES

# Specifications (not tests)

### STS-1 Mission Facts — Columbia

Commander: John Young  
Pilot: Robert Crippen  
Mission Duration — 54 hours, 21  
57 seconds  
Miles Traveled — Approximately  
1,074,567 statute miles  
Orbits of Earth — 36

### STS-2 Mission Facts — Columbia 1981

Commander: Joe Engle  
Pilot: Richard Truly  
Mission Duration — 54 hours, 24 minutes,  
4 seconds  
Miles Traveled — Approximately 933,757 nautical  
miles (1,074,567 statute miles)  
Orbits of Earth — 3  
Cargo Weight — Approximately 8,771 kilograms  
(19,388 pounds)

### STS-3 Mission Facts — Columbia — March 22-30, 1982

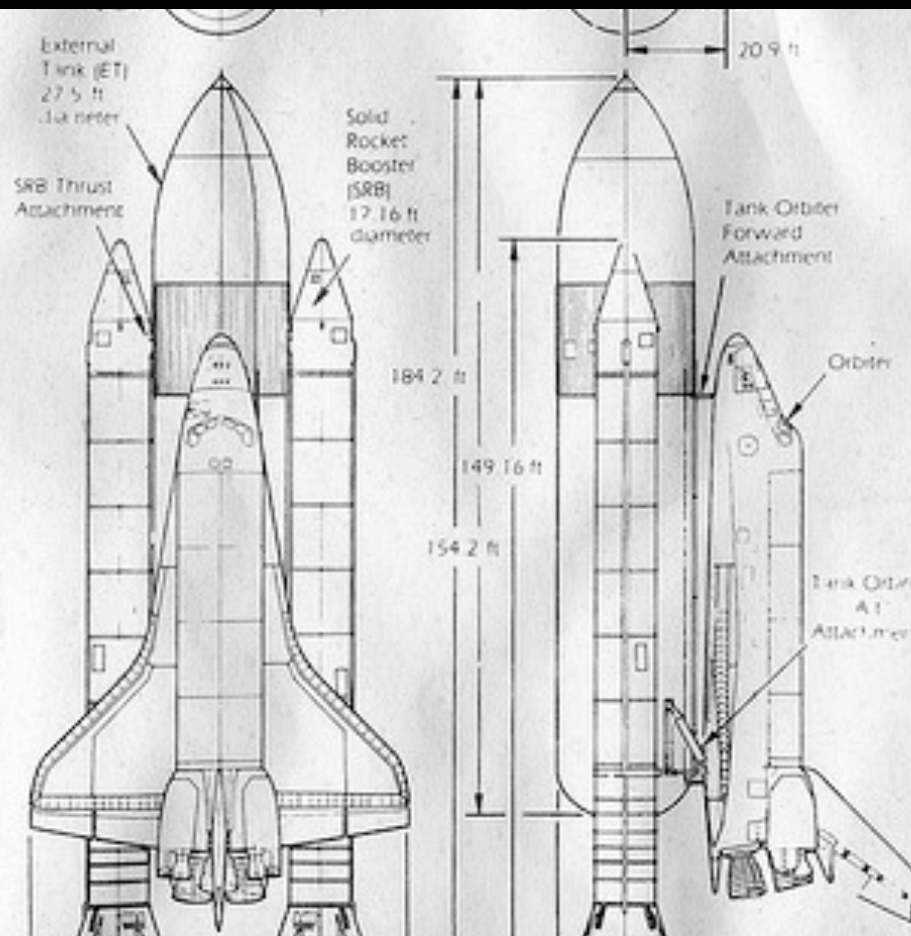
Commander: Jack Lousma  
Pilot: Gordon Fullerton  
Mission Duration — 192 hours (8 days), 6 minutes,  
9 seconds  
Miles Traveled — Approximately 3.9 million nautical  
miles (4.4 million statute miles)  
Orbits of Earth — 130  
Cargo Weight — Approximately 9,658 kilograms  
(21,293 pounds)

### STS-4 Mission Facts — Columbia — June 27, July 4, 1982

Commander: Ken Mattingly  
Pilot: Henry Hartsfield  
Mission Duration — 168 hours (7 days), 1 hour,  
10 minutes, 43 seconds  
Miles Traveled — Approximately 2.9 million nautical  
miles (3.3 million statute miles)  
Orbits of Earth — 112 orbits

### STS-5 Mission Facts — Columbia — November 11-16, 1982

Commander: Vance Brand  
Pilot: Robert Overmyer  
Mission Specialist: Joseph Allen  
Mission Specialist: William Lenoir  
Mission Duration — 120 hours (5 days), 2 hours,  
15 minutes, 29 seconds  
Miles Traveled — 1.5 million nautical miles (1.8 million  
statute miles)  
Orbits of Earth — 81  
Cargo Weight — Approximately 14,974 kilograms



### SPACECRAFT

Spacecraft, a double delta-winged airplane developed by Rockwell International's Space Shuttle Company, is designed to perform a minimum of 30 orbits of Earth orbit on a quick-turnaround basis.

Orbits of the Spacecraft are: Crew Compartment arrangement for four crew members; flight controls for pilot and co-pilot on the main deck; area in mid deck, which also houses the main engine compartment; and operating spacecraft, will be able to accommodate additional specialists/scientists.

Measures 15 ft in diameter by 60 ft in length. Capacity for up to 65,000 pounds of cargo.

Orbital Maneuvering/Reaction Control System — Orbital Maneuvering System (OMS) has two 6,000-pound thrust engines in pods, one on each side of the spacecraft's vertical stabilizer. Reaction Control System (RCS) has 38 (thirty-eight) 870-pound thrust engines in six 24-pound vernier thrusters. Fourteen of the larger RCS engines are in the spacecraft's nose and 24 are on the side. 12 in each OMS pod. Two of the smaller thrusters are in the forward pod and four on the aft end, two in each OMS pod.

Thermal Protection — Consists of a silica fiber based, high temperature and low temperature reusable surface insulation in addition to a coated Nomex felt over a majority of the craft and a reinforced carbon-carbon composite for the nose and wing leading edges. Insulation materials on the leading edge of the wing and nose of the spacecraft must be able to withstand temperatures up to 2,300°F on reentry from orbital flights and be reusable.

### SPACE SHUTTLE FACTS (weights approximate)

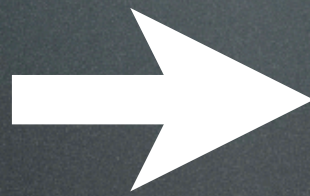
LENGTH	
SYSTEM	184.2
ORBITER	122.2
HEIGHT	
SYSTEM	75.6
ORBITER	56.6
WINGSPAN	
ORBITER	78.06
WEIGHT	
GROSS LIFT-OFF	4.5 million
ORBITER LANDING	varies dependent upon mission in thousand
THRUST	
SOLID ROCKET BOOSTERS (SRB) (2)	2.9 million lb of thrust each at sea level
ORBITER MAIN ENGINES (3)	393,800 lb of thrust each at sea level
LOAD CAPACITY	
DIMENSIONS	60 ft long, 15 ft in diameter



TDD → BDD



Testing  
Code



Specifying  
Behavior



RSpec != Specifying  
Behavior



# Two Questions



(1)

If I wanted to use this  
software in my project,  
what behaviors are  
important to me?



(1)

Necessary



(2)

Could I write this  
software from scratch  
using only the tests/  
specs as guidance?



(2)

Sufficient



Things that are  
Important ...



public methods  
(names, args, contract)



# public protocols



Things that are  
**NOT** Important ...



# private methods



# Ancillary Objects





# Summary

Copyright ©2004 Mayang Admin. See <http://www.mayang.com/textures/>



# (1) Tests are Code

Treat them with the same respect  
as the rest of your source code.



# (2) Tests are Specifications

Focus on the **What**,  
Not the **How**



# Questions?

Jim Weirich  
Chief Scientist / EdgeCase  
jim@edgecase.com  
@jimweirich



(photo by James Duncan Davidson)



# Image Attributions

cables: Scott the (angrykeyboarder on Flickr)

snail: <http://photozou.jp/photo/show/38290/21923871>

data center: Christopher Bowns (cbowns on Flickr)

report card: (ambo who? on Flickr)

giraffe: (Kurt Thomas Hunt on Flickr)

custom guitar: (The Creamery on Picasa)

escher mirror: (Bert K on Flickr)

privacy: Rob Pongsajapan (rpongsaj on Flickr)

russian dolls: (frangipani photograph on Flickr)

shuttle specs: Tom Peck (ThreadedThoughts on Flickr)

bubble wrap: GNU Free Documentation License.

questions: (Rock Alien on Flickr)



# License



Attribution-NonCommercial-ShareAlike 2.0